SOUND SOURCE TRACKING USING MICROPHONE ARRAYS AND DEEP LEARNING

Eduardo Alves da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Luiz Wagner Pereira Biscainho

Rio de Janeiro
Setembro de 2022

SOUND SOURCE TRACKING USING MICROPHONE ARRAYS AND DEEP LEARNING

Eduardo Alves da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Luiz Wagner Pereira Biscainho

Aprovada por: Prof. Luiz Wagner Pereira Biscainho
Prof. José Gabriel Rodriguez Carneiro Gomes
Prof. Diego Barreto Haddad

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2022

Alves da Silva, Eduardo

Sound source tracking using microphone arrays and deep learning/Eduardo Alves da Silva. – Rio de Janeiro: UFRJ/COPPE, 2022.

XII, 61 p.: il.; 29, 7cm.

Orientador: Luiz Wagner Pereira Biscainho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2022.

Referências Bibliográficas: p. 54 – 61.

1. Aprendizagem Profunda. 2. Processamento de Sinais de Arranjos de Microfone. 3. Rastreamento de Fontes Sonoras. I. Pereira Biscainho, Luiz Wagner. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

To my sleep. Only I know how
much you have suffered
throughout these years.

Agradecimentos

This section is written in Portuguese. If you are reading this document only for the mathematical/computational stuff, you can skip this section.

Em primeiro lugar, gostaria de agradecer aos meus pais, Maria das Graças e Eduardo Jorge, por todo amor e por todo o apoio incondicional que recebi durante minha formação. Vocês me ajudaram a entender o valor que a educação tem, e que eu nunca devo medir esforços quando se trata de estudos.

Gostaria de agradecer ao meu irmão, Fernando Alves da Silva. Sem você, esse mundo não teria graça nenhuma.

À minha namorada, Maria Eduarda, pelo carinho e pelo apoio, principalmente naqueles momentos quando duvidei da minha capacidade. Só você é capaz de tornar os dias mais difíceis nos melhores sem nenhum esforço. Muito obrigado por ser a minha fonte sonora favorita.

Ao meus amigos do Grupo de Processamento de Áudio e do Laboratório de Sinais, Multimídia e Telecomunicações, por todas as conversas, discussões e pelos incontáveis cafés.

Aos meus amigos de fora da universidade, que também estiveram presentes em minha vida e me ajudaram a tirar da cabeça todas as equações e algoritmos nos momentos (quase) sempre apropriados.

Aos professores do Programa de Engenharia Elétrica que contribuíram, de alguma forma, a minha formação.

Ao meu orientador, Luiz Wagner, por uma infinidade de coisas. Por todas as horas me explicando equações, teoremas, algoritmos e técnicas, pelas conversas descontraídas, sejam elas nas caminhadas pelo Centro de Tecnologia ou no Zoom, e por ser um exemplo de profissional. Se um dia eu me tornar um engenheiro com metade da sua capacidade, ficarei satisfeito.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

RASTREAMENTO DE FONTES SONORAS POR MEIO DE ARRANJOS DE MICROFONES E APRENDIZAGEM PROFUNDA

Eduardo Alves da Silva

Setembro/2022

Orientador: Luiz Wagner Pereira Biscainho

Programa: Engenharia Elétrica

A irrupção de aplicações como alto-falantes inteligentes, robótica para interação humana, dentre outras, gerou uma pressão pelo desenvolvimento de técnicas mais avançadas de processamento de sinais de áudio e aprendizagem de máquina. Dezenas de sistemas capazes de realizar localização e rastreamento de fontes sonoras surgiram. Neste trabalho, apresentamos uma técnica nova, chamada Spectral Cross3D, inspirada em técnicas preexistentes de processamento de sinais de arranjos de microfones e aprendizagem profunda, para o rastreamento de fontes acústicas. A partir da combinação do clássico método de estimação de direção de chegada MUSIC com uma rede neural convolucional, foi possível obter uma nova abordagem capaz de obter resultados competitivos com arquiteturas do estado da arte.

Esse trabalho se divide em duas partes. Na primeira, a teoria acerca do processamento de sinais de arranjos de microfones é introduzida, juntamente com métodos de localização e rastreamento de fontes sonoras clássicos. Em seguida, o sistema proposto é apresentado, os experimentos conduzidos para sua avaliação são explicados e, por fim, os resultados são discutidos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SOUND SOURCE TRACKING USING MICROPHONE ARRAYS AND DEEP LEARNING

Eduardo Alves da Silva

September/2022

Advisor: Luiz Wagner Pereira Biscainho

Department: Electrical Engineering

The explosion of applications such as intelligent speakers, human-robot interaction and many others have driven the development of more advanced techniques of audio signal processing and machine learning. Dozens of systems capable of performing sound source localization and tracking were brought to light. In this work, we present a novel technique, called Spectral Cross3D, inspired by existing techniques of microphone array signal processing and deep learning for acoustic source tracking. From the combination of the classical MUSIC direction-of-arrival estimator with a convolutional neural network, it was possible to obtain a new approach that achieves results competitive with those of state-of-the-art architectures.

This work is divided into two parts. First, the theory around microphone array signal processing is introduced, along with the classical methods for sound source localization and tracking. Next, the proposed system is presented, the experiments designed to evaluate it are explained and, finally, the results of these experiments are discussed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ability to localize multiple sound sources is innate to humans. Being able to identify, through audition, the correct position of prey, predators or any source of imminent danger was crucial for the success of our species. Also, whenever we see ourselves in an environment where multiple speakers are present, spatial information is often necessary to perform proper sound source segregation. Thus, it makes sense that there is a strong desire to replicate this capacity in computational systems. This has led to studies in the area of microphone array signal processing [1, 2], with the goal of performing sound source localization. Moreover, applications such as human-robot interaction [3], acoustic navigation [4], audio based surveillance [5], and intelligent speakers [6] started to be discussed and developed.

When we consider scenarios of multiple sound sources, moving sound sources, moving microphones, or high levels of noise and reverberation, it becomes clear that the problem at hand is rather complex. The recent success of deep learning [7] for supervised tasks has made it clear that exploring their applicability for this task is a viable path to tackle more challenging situations. As a result, many different approaches using deep learning for acoustic source tracking were developed.

1.1 Objectives

This project has two main goals. The first objective is to study both classical and more recent machine learning based approaches to sound source localization and tracking. More specifically, this text is intended to become a gentle introduction to these topics, for people looking forward to working with acoustic localization. The second objective is to develop a novel system capable of performing sound source tracking, specifically for the case of single speech source. It is desirable that the system is evaluated on a public dataset, and have its results compared with some state-of-the-art system from the literature.

1.2 Materials

All the systems in this work were implemented using the Python programming language. Specifically, the PyTorch framework was intensively used, with many other scientific computing libraries. Network training was performed in an NVIDIA Titan Xp[1] graphical processing unit. The LaTeX typesetting software was used for documentation.

1.3 Outline

After this Introduction, the work is organized as follows. The basic signal model used in most of acoustic localization literature is described in Chapter 2. The main solutions for sound source localization and tracking are described in Chapters 3 and 4, respectively. Then, our system is introduced in Chapter 5. We describe the training and test procedures in Chapter 6, where we also discuss the results of the experiments. Final conclusions and proposals for future works are presented in Chapter 7.

[1] `https://www.nvidia.com/en-us/titan/titan-xp/`

Chapter 2

Signal Model

In this section, we will introduce the signal modeling necessary for a complete understanding of sound source localization and tracking techniques.

2.1 Microphone array

A microphone array consists of a set of M microphones arranged in some geometry, as depicted in Figure 2.1. The sound captured by the microphone indexed by $m \in \{1, 2, \ldots, M\}$ is a digital signal sampled at frequency F_s (measured in Hz), denoted by $y_m[n]$. The following simplifications are adopted for the microphone array and the signal captured by the microphones:

- Each microphone is subjected to zero-mean independent white noise, denoted by $v_m[n]$.
- Every microphone is omnidirectional, i.e. it does not favor sound coming from any specific direction[1].
- The array mounting surface does not influence the sound captured by the microphones, i.e. diffraction effects due to microphone mounting can be ignored[2].

2.1.1 Single Sound Source

When a sound source is emitting an acoustic signal $s(t)$ in an anechoic environment, the sound captured by microphone m is given by

$$x_m(t) = \alpha_m s(t - t_m), \tag{2.1}$$

[1] This is generally not the case for the Ambisonics array framework, which will not be discussed in this work.

[2] This is generally not the case for the spherical array framework, which will not be discussed in this work.

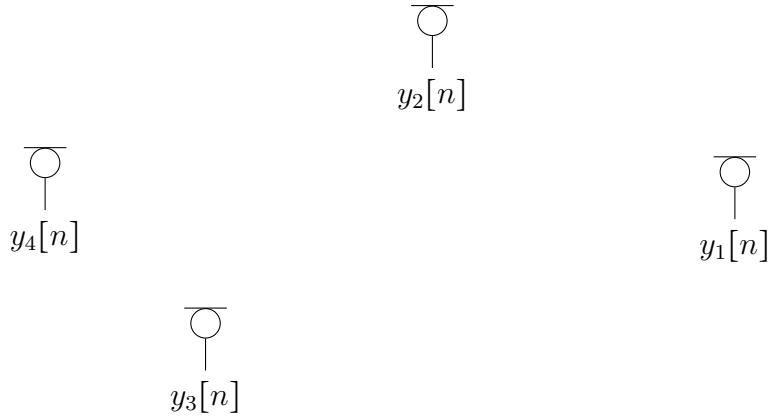


Figure 2.1: Microphone array, comprised of 4 microphones.

where

$$\alpha_m \propto \frac{1}{\|\mathbf{r}_s - \mathbf{r}_m\|} \tag{2.2}$$

is the sound wave attenuation,

$$t_m = \frac{\|\mathbf{r}_s - \mathbf{r}_m\|}{c} \tag{2.3}$$

is the time delay; $\|\cdot\|$ denotes the Euclidean norm of a vector, $\mathbf{r}_s$ is the source position, $\mathbf{r}_m$ is the position of the m-th microphone (both in Cartesian coordinates) and c is the speed of sound.

Since measuring the absolute time delay is impractical, it is common to reparameterize Equation (2.1) as

$$x_m(t) = \alpha_m s(t - t_r - \tau'_{rm}), \tag{2.4}$$

where the r indexes the reference microphone and

$$\tau'_{ij} = t_j - t_i \tag{2.5}$$

is the time delay difference, commonly referred to as the time difference of arrival (TDOA).

As we will only work with discrete-time signals, and to keep consistence with the related literature, we will define τ_{ij} as the value of τ'_{ij} measured in samples taken at sampling frequency F_s:

$$\tau_{ij} = F_\mathrm{s}\tau'_{ij} = F_\mathrm{s}\frac{\|\mathbf{r}_s - \mathbf{r}_i\| - \|\mathbf{r}_s - \mathbf{r}_j\|}{c} \tag{2.6}$$

As will be seen, Equation (2.6) is extremely useful, as it maps the position of the

sound source to the expected TDOA of any two microphones. Since the evaluation of τ_{ij} can be made without prior knowledge of t_r, it is well suited for sound source localization methods.

It should be noted that the discretization process will lead to approximation errors in the final estimate. In this work, we will assume that F_s is high enough that we can ignore this error.

A common simplification made to the aforementioned model is to assume the distance between the sound source and any microphone in the array is a few orders of magnitude greater than the distance between any two microphones in the array. This leads to the so-called far field model, in which the following assumptions are valid:

1. All attenuation coefficients α_i, $i = 1, 2, \ldots, M$, are equal;
2. The TDOA depends only on the azimuth angle θ and the elevation angle ϕ, depicted in Figure 2.2.

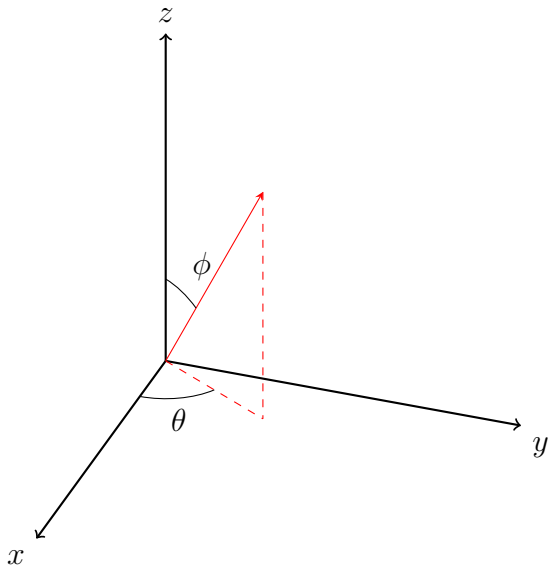


Figure 2.2: Definition of the azimuth (θ) and elevation (ϕ) angles.

Therefore, throughout this work, we will assume that, unless stated otherwise, $\alpha_m = \alpha$, $m = 1, 2, \ldots, M$. Also, taking in account the previous comments, we can simplify our modeling of the TDOA and rewrite Equation (2.6) as

$$\tau_{ij} = \frac{1}{c}\boldsymbol{\zeta}^T(\phi, \theta)(\mathbf{r}_j - \mathbf{r}_i), \tag{2.7}$$

where [8]

$$\boldsymbol{\zeta}^T(\phi, \theta) = \begin{bmatrix} \sin\phi\cos\theta & \sin\phi\sin\theta & \cos\phi \end{bmatrix}^T. \tag{2.8}$$

Finally, we can write the signal that is captured by a microphone in the presence of a sound source as

$$y_m[n] = x_m[n] + v_m[n] = \alpha s[n - F_s t_r - \tau_{rm}] + v_m[n]. \tag{2.9}$$

Also, in some cases it will be convenient to represent the output of each microphone as an element of column vector

$$\mathbf{y}[n] = \begin{bmatrix} y_1[n] \\ y_2[n] \\ \vdots \\ y_M[n] \end{bmatrix} = \begin{bmatrix} x_1[n] \\ x_2[n] \\ \vdots \\ x_M[n] \end{bmatrix} + \begin{bmatrix} v_1[n] \\ v_2[n] \\ \vdots \\ v_M[n] \end{bmatrix} = \mathbf{x}[n] + \mathbf{v}[n]. \tag{2.10}$$

Uniform linear microphone array

A particularly interesting case of microphone array is the so-called uniform linear microphone array (ULA), where the microphones are uniformly spaced in a linear configuration along the x-axis as depicted in Figure 2.3.

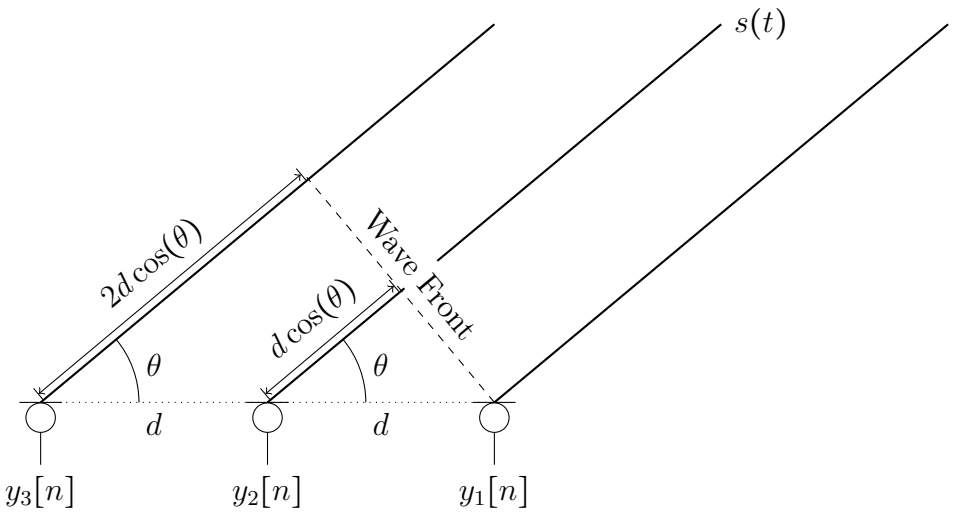


Figure 2.3: Diagram of a wave front hitting a 3-microphone uniform linear array.

This array configuration is specially useful in the far field case. Consider a sound source $s(t)$ at a great distance from (but at the same height as) a uniform linear array, where the spacing between two adjacent microphones is d. In this scenario, the spherical acoustic wave is seen as a planar wave[3], which impinges the array at an angle θ with respect to the line connecting the microphones. Therefore, the difference between the distance traveled by the wave arriving at two adjacent microphones is

[3]This is another way of interpreting the second assumption made previously.

constant (as indicated in Figure 2.3), leading to [1]

$$\tau_{i,i+1} = \frac{d\cos(\theta)}{c}. \tag{2.11}$$

Taking microphone 1 as the reference, one can simplify equation (2.7) as

$$\tau_{1m} = (m-1)\tau_{12} = (m-1)\tau. \tag{2.12}$$

This simple formulation of the TDOA makes the ULA a good choice for most source localization methods.

2.1.2 Multiple sources

Whenever there are multiple sources present, one must extend the model presented in equation (2.9) to take them all into account. In an acoustic scene comprised of L sound sources, each emitting the signal labeled as $s_l(t)$, all these signals are added at the microphone, leading to

$$x_m[n] = \sum_{l=1}^{L} \alpha_l s_l[n - F_s t_{l,r} - \tau_{l,mr}], \tag{2.13}$$

where $t_{l,r}$ and $\tau_{l,ij}$ are defined exactly as (2.3) and (2.6) respectively, according to the position of source l.

2.1.3 Reverberation

In all these settings, we considered only the anechoic environment, i.e. either the sound source is in the open field or in an enclosed space where the sound waves are absorbed by the walls, ceiling, floor and any object inside the room. In reality, each time the travelling sound wave hits an object only a fraction of the energy gets absorbed, and the remaining energy acts as a reflected wave. The ratio between absorbed and reflected energy depends (mostly) on the material of that object. A full analysis of the reflection process can be found in Kuttruff et al [9]. In the present work, we will consider only the case of specular reflections, i.e. reflections on perfectly smooth surfaces.

Consider that a sound wave propagating from a source hits W surfaces, each with a reflection coefficient β_w, and K samples are taken upon arrival at a microphone. The signal produced by that sound wave is given by

$$s[n-K]\prod_{w=1}^{W}\beta_w. \tag{2.14}$$

Along with the wave that travels along the direct path from the source to the array, an infinite number of sound waves hit the array with different delays and different attenuation constants. This is equivalent to passing the source signal through a linear, time-invariant, spatial filter. In other words, we can consider that if the source is stationary in space, then equation (2.1) can be extended to incorporate multiple reflections as

$$x_m[n] = (g_m * s)[n], \tag{2.15}$$

where $g_m[n]$ represents the impulse response of the room (room impulse response, or RIR) relating to the position of the source at time n and the position of microphone m.

The standard modeling approach considers that after a certain number K of samples, the energy of the waves that reach the microphone is too small and can be disregarded. This allows one to use system identification techniques to model the RIR. However, a common problem of this model is that it requires a large value of K to attain a reasonable performance.

2.2 Sound source localization

The problem of sound source localization (SSL) using microphone arrays can now be properly defined as follows. Given a set of samples of the output $\mathbf{y}$ belonging to a microphone array with known geometry, produce an estimate $\hat{\mathbf{r}}_s$ of the position of sound sources active in the acoustic scene, emitting the signals $s_l(t), l = 1, \ldots, L$.

A complete localization procedure aims to identify three spatial coordinate values. However, it is common to reduce the problem to simply identifying the angular coordinates θ and ϕ of the source. Depending on the array geometry and on the end goal, one can go a step further and take only the azimuth as the desired coordinate. In either case, the simplified problem is called Direction of Arrival (DOA) determination.

It must have been clear that the TDOA between multiple microphones plays a significant role in the SSL problem. Equations (2.6), (2.7) and (2.11) show how one can map the sound source position onto a equivalent set of TDOA values. We can point out two different types of solutions [8]:

1. Directly estimating the TDOA between every pair of microphones, and through a reverse mapping, finding the most likely source position according to some criterion (e.g. maximum likelihood). This is sometimes called the two-step approach.

2. Given a set of candidate positions for the sound source, selecting the optimum

solution by either maximizing or minimizing (depending on the context) some suitable spatio-temporal statistic.

Since the mapping transformation of the TDOA from the source position is nonlinear, small errors in the estimation may lead to significantly noisy position estimates. In contrast, the second approach presented involves the evaluation of a function which, as we will see in Chapter 3, requires the calculation of all the TDOAs associated with the candidate positions in the whole search space, which can be computationally demanding, but gives a more robust position estimate.

2.3 Sound source tracking

The extension of the SSL problem for non-stationary sound sources leads to the problem of sound source tracking (SST), which we define as follows. Given a set of observations of the output signal $\mathbf{y}[n]$ produced by a microphone array with known geometry, estimate the trajectories of a set of sound sources $\hat{\mathbf{r}}_{s_l}[j]$, $l = 1, \ldots, L$.

As we shall see in Chapter 3, most methods require a set of W contiguous samples (often referred to as a window or a frame) to produce a single estimate $\hat{\mathbf{r}}_{s_l}[j]$. So, it will often be the case that the observations vector $\mathbf{y}[n]$ and the positions estimates have two different time scales. To remove any ambiguity, we will always use the index j when referring to the frame-based time scale, and n for the sample-based one.

In general, some sort of evolution model is assumed, aiming to predict the future position of the sound source given present and previous estimates. If there is no need for real time tracking, one can combine the previous approach with a retrodiction model, which attempts to model past source positions given present and future observations and estimates.

In the present work, we will restrict ourselves to the task of real-time tracking of a single sound source.

Chapter 3

Classical Solutions for Sound Source Localization

In this chapter, we are going to see some examples of solutions to the SSL problem. We start discussing techniques of TDOA estimation (often called time delay estimation, or TDE techniques), necessary for the two-step approach discussed previously. We follow it with a presentation of the steered response power method and the multiple signal classifier method, both centered around the use of a spatiotemporal statistic for localization. We finish the chapter presenting neural networks as a newer class of solutions to SSL.

Throughout the chapter, we will deal with the following setup: an acoustic source positioned at $\mathbf{r}_s$ emitting a certain signal $s[n]$. The microphone array is comprised of M microphones, each at a known position $\mathbf{r}_m$. Since we have defined that the noise captured by each microphone is white, they all have the same cross-correlation, given by $r_{vv}^{CC}[k] = \sigma_v^2 \delta[k]$[1].

3.1 TDOA Estimation

In order to estimate the TDOA, we are going to make use of the second order statistical moments of the microphone signals. More specifically, the time cross-correlation is going to be useful, as we shall see briefly.

3.1.1 Cross-Correlation

For instance, assume there are two microphones with signals $y_1[n]$ and $y_2[n]$ in an anechoic acoustic scene with a sound source $s[n]$ present, which is emitting white

[1] $\delta[k] = \begin{cases} 1, & k = 0 \\ 0, & \text{elsewhere} \end{cases}$

noise with zero mean and auto-correlation $r_{ss}^{CC}[k] = \sigma_s^2\delta[k]$. Then, (2.9) leads to

$$y_1[n] = \alpha s[n - F_s t_1] + v_1[n] \tag{3.1}$$

$$y_2[n] = \alpha s[n - F_s t_1 - \tau_{12}] + v_2[n]. \tag{3.2}$$

Given that the noise signals are uncorrelated with each other and with the source signal, the cross-correlation $r_{y_2y_1}^{CC}[k]$ of the two signals is given by

$$r_{y_2y_1}^{CC}[k] = \mathbb{E}[y_2[n]y_1[n-k]] \tag{3.3}$$

$$= \alpha^2 r_{ss}^{CC}[k - \tau_{12}] \tag{3.4}$$

$$= \alpha^2\sigma_s^2\delta[k - \tau_{12}], \tag{3.5}$$

which presents a peak at the value of $k = \tau_{12}$. Even in the case where the sound source is not white-noise, it is well known that (3.4) will have its maximum value in that same position [10]. Thus, if one is able to calculate the cross-correlation of the signal at microphones i and j, it is possible to identify the TDOA τ_{ij} through

$$\hat{\tau}_{ij}^{CC} = \arg\max_k r_{y_iy_j}^{CC}[k]. \tag{3.6}$$

Since $r_{y_iy_j}^{CC}[k]$ would require an expectation amongst all possible realizations of y_i and y_j, in practice an estimate $\hat{r}_{y_iy_j}^{CC}[k]$ is calculated by averaging along the temporal dimension. Considering a time window of W samples, comprised of $\{y_i[n], y_i[n+1], \ldots, y_i[n+W-1]\}$ and $\{y_j[n], y_j[n+1], \ldots, y_j[n+W-1]\}$, it is possible to obtain asymptotically unbiased estimations with

$$\hat{r}_{y_iy_j}^{CC}[k] = \begin{cases} \frac{1}{W}\sum_{w=0}^{W-k-1} y_i[n+w]y_j[n+w-k], & k \geq 0 \\ \hat{r}_{y_iy_j}^{CC}[-k], & k < 0. \end{cases} \tag{3.7}$$

A closer look at Equation (3.7) reveals that one can perform the same operation in the frequency domain [11]. Thus, if the discrete Fourier transforms (DFTs) of y_i and y_j are given by $Y_1(\omega)$ and $Y_j(\omega)$, respectively, one can write

$$\hat{r}_{y_iy_j}^{CC}[k] = \mathcal{F}^{-1}\left[Y_i(\omega)Y_j^*(\omega)\right] = \int_{-\infty}^{\infty} Y_1(\omega)Y_2^*(\omega)e^{\jmath\omega n}\mathrm{d}\omega. \tag{3.8}$$

3.1.2 Generalizations to the Cross-Correlation

Equation (3.8) can be generalized by introducing a factor $\Psi(\omega)$ on its right-hand side, as

$$\hat{r}_{y_iy_j}^{GCC}[k] = \int_{-\infty}^{\infty} \Psi(\omega)Y_1(\omega)Y_2^*(\omega)e^{\jmath\omega n}\mathrm{d}\omega. \tag{3.9}$$

This leads to the so-called generalized cross correlation[12] (GCC), where one can use different weights for every frequency component, in order to attain a better estimate for the TDOA.

There are many different possible choices for $\Psi(\omega)$, which leads to a family of estimators [13–16]. Here, we will bring into focus the phase transform (PHAT).

Phase Transform

The phase transform[1] arises from the observation that the TDOA information is conveyed by the phase rather than the amplitude of the signals. Thus, to avoid the fluctuations imposed by different absolute values, we choose $\Psi(\omega) = (|Y_i(\omega)Y_j^*(\omega)|)^{-1}$, which yields

$$\hat{r}_{y_i y_j}^{PHAT}[k] = \int_{-\infty}^{\infty} \frac{Y_1(\omega)Y_2^*(\omega)}{|Y_1(\omega)Y_2^*(\omega)|} e^{\jmath\omega n} \mathrm{d}\omega = \int_{-\infty}^{\infty} e^{\jmath(\angle Y_i(\omega) - \angle Y_j(\omega))} e^{\jmath\omega n} \mathrm{d}\omega. \tag{3.10}$$

Due to its robustness, the phase transform has found large applicability in different schemes of source localization.

3.1.3 Least Squares Estimation of Source Position

Having estimated the values of all the different TDOAs, the next step is to identify the position of the source. There are many different solutions for this task in the literature, and here we will present one such approach, based on the least squares (LS) spherical error minimization [17].

For the sake of simplicity, we present a version of the algorithm that uses only the values of τ_{m1}, $m = 2, \ldots, M$. The extension to the complete version is direct. Also, without loss of generality, we are going to assume that the microphone of index 1 is positioned at the origin of the Cartesian plane, i.e. $\mathbf{r}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.

Firstly, we introduce the quantity D_m, which is the distance from the source to the m-th microphone. By definition, $D_1 = \|\mathbf{r}_s\|$. For $m = 1, \ldots, M$, we have

$$D_m = \|\mathbf{r_s} - \mathbf{r_m}\| = D_1 + d_{m1}, \tag{3.11}$$

where d_{m1} is defined as the difference $D_m - D_1$. Given (2.6), it is possible to estimate d_{m1} as

$$\hat{d}_{m1} = \frac{c}{F_s} \hat{\tau}_{m1}, \tag{3.12}$$

providing the estimate $\hat{D_m} = \|\mathbf{r}_s\| + \hat{d}_{m1}$. Also, equation (3.11) leads to:

$$D_m^2 = \|\mathbf{r}_s\|^2 - 2\mathbf{r}_s^T\mathbf{r}_m + \|\mathbf{r}_m\|^2. \tag{3.13}$$

Combining (3.13) with the definition of $\hat{D}_m$, we define the squared spherical error $e_{\text{sp},m}$ as a function of a candidate position $\hat{\mathbf{r}}_s$, where

$$\begin{aligned} e_{\text{sp},m}(\hat{\mathbf{r}}_s) &= \frac{1}{2}(\hat{D}_m^2 - D_m^2) \\ &= \hat{d}_{m1}\|\mathbf{r}_s\| + \mathbf{r}_s^T\mathbf{r}_m - \frac{1}{2}(\|\mathbf{r}_m\|^2 - \hat{d}_{m1}^2). \end{aligned} \tag{3.14}$$

The spherical error receives its name due to the fact that it evaluates the mismatch between the estimated radial distances between the microphone m and the sound sources and their true values. Equation (3.14) can be extended by stacking all errors $e_{\text{sp},m}$, $m = 2, \ldots, M$ in a vector notation, as

$$\mathbf{e}_{\text{sp}}(\mathbf{r}_s) = \mathbf{A}\boldsymbol{\theta} - \boldsymbol{\xi}, \tag{3.15}$$

where we have defined

$$\mathbf{A} = [\mathbf{S}\,|\,\mathbf{d}] = \left[\begin{array}{c|c} —\mathbf{r}_2^T— & \hat{d}_{21} \\ \vdots & \vdots \\ —\mathbf{r}_M^T— & \hat{d}_{M1} \end{array}\right] \quad \boldsymbol{\theta} = \begin{bmatrix} | \\ \mathbf{r}_s \\ | \\ \|\mathbf{r}_s\| \end{bmatrix} \quad \boldsymbol{\xi} = \begin{bmatrix} \frac{1}{2}(\|\mathbf{r}_2\|^2 - \hat{d}_{21}^2) \\ \vdots \\ \frac{1}{2}(\|\mathbf{r}_M\|^2 - \hat{d}_{M1}^2) \end{bmatrix}. \tag{3.16}$$

Thus, our goal is to minimize $\|\mathbf{e}_{\text{sp}}\|^2$. One such way to perform this is the so called spherical intersector (SX) [18]. It comes from the observation that Equation (3.15) would have a linear dependence on $\mathbf{r}_s$ if the value of $\|\mathbf{r}_s\|$ were known and vice-versa. Thus, to implement the SX estimator, we assume initially that the value of $\|\mathbf{r}_s\|$ is known. Then, the estimate $\hat{\mathbf{r}}_s$ is given by [19]

$$\hat{\mathbf{r}}_s^{\text{LS-SP}} = \mathbf{S}^\dagger(\boldsymbol{\xi} - \|\mathbf{r}_s\|\mathbf{d}), \tag{3.17}$$

where $\mathbf{S}^\dagger = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}$. Then, observing that $\|\mathbf{r}_s\|^2 = \mathbf{r}_s^T\mathbf{r}_s$, with Equation (3.17) we get

$$\|\mathbf{r}_s\|^2 = \left[\mathbf{S}^\dagger(\boldsymbol{\xi} - \|\mathbf{r}_s\|\mathbf{d})\right]^T\left[\mathbf{S}^\dagger(\boldsymbol{\xi} - \|\mathbf{r}_s\|\mathbf{d})\right] \implies a_1\|\mathbf{r}_s\|^2 + a_2\|\mathbf{r}_s\| + a_3 = 0, \tag{3.18}$$

where $a_1 = 1 - \|\mathbf{S}^\dagger\mathbf{d}\|^2$, $a_2 = 2\boldsymbol{\xi}^T\mathbf{S}^{\dagger T}\mathbf{S}^\dagger\mathbf{d}$ and $a_3 = -\|\mathbf{S}^\dagger\boldsymbol{\xi}\|^2$. Equation (3.18) can be solved in order to find the value of $\|\mathbf{r}_s\|$. If there is only one real and positive solution, it is taken and used in (3.17) to find the source position. If there are two real positive solutions and neither can be discarded by some criterion (e.g. being outside the region of interest) or if none of the solutions is a positive real number, the method fails to present a location. If that happens, one must apply other source position estimation methods [17].

3.2 Steered Response Power

The intuition behind the steered response power (SRP) methods is to use a tempo-spatial filter (also commonly known as beamformer) to extract only the sound coming from one specific direction and evaluate its power. A grid search is then performed to generate a map of the acoustic scene, with high energy points indicating a high probability of having a sound source at that position. Many methods apply this principle, using different types of beamformers as the main filter [20, 21], and the SRP uses the delay and sum (DS) technique [1].

In order to explain its fundamentals, it will be useful to define $\tau_{1m}(\mathbf{r})$ as the expected TDOA value considering the source to be positioned at $\mathbf{r}$. The DS beamformer time aligns the different microphones, and averages their signals in order to attenuate noise. Denoting z_{DS} as the output, this is equivalent to

$$z_{DS}[n, \mathbf{r}] = \frac{1}{M} \sum_{m=1}^{M} y_m[n + \tau_{1m}(\mathbf{r})] \tag{3.19}$$

The SRP method consists of evaluating the power of $z_{DS}[n, \mathbf{r}]$, given by

$$P_{\text{SRP}}(\mathbf{r}) = \int_{-\infty}^{\infty} |Z_{DS}(\mathbf{r}, \omega)|^2 \mathrm{d}\omega, \tag{3.20}$$

where $Z_{DS}(\mathbf{r}, \omega)$ is the Fourier transform of $z_{DS}[n, \mathbf{r}]$, in a grid of possible values of $\mathbf{r}$. This generates a heatmap where points that are associated with higher power measures have a high probability of containing a sound source. Thus, to identify the position of a sound source, the maximum of (3.20) needs to be evaluated, leading to

$$\hat{\mathbf{r}}_s^{\text{SRP}} = \arg_{\mathbf{r}} \max P_{\text{SRP}}(\mathbf{r}). \tag{3.21}$$

In [2], a more general formulation was presented, which relates the SRP method and the GCC equation (3.9), and it is given by

$$P_{\text{SRP-GCC}}(\mathbf{r}) = \sum_{m=1}^{M} \sum_{l=1}^{M} \int_{-\infty}^{\infty} \Psi_{lm}(\omega) Y_l(\omega) Y_m^*(\omega) e^{\jmath \omega \tau_{ml}(\mathbf{r})} \mathrm{d}\omega \tag{3.22}$$

$$= \sum_{m=1}^{M} \sum_{l=1}^{M} \hat{r}_{y_m y_l}^{GCC}[\tau_{ml}(\mathbf{r})]. \tag{3.23}$$

Just as we did in Section 3.1.1, we can use the PHAT to generate more robust power maps and increase localization performance. This leads to the so-called SRP-PHAT method, given by

$$\hat{\mathbf{r}}_s^{\text{SRP-PHAT}} = \arg_{\mathbf{r}} \max P_{\text{SRP-PHAT}}(\mathbf{r}) = \arg_{\mathbf{r}} \max \sum_{m=1}^{M} \sum_{l=1}^{M} \hat{r}_{y_m y_l}^{PHAT}[\tau_{ml}(\mathbf{r})]. \tag{3.24}$$

The main issue associated with the SRP-PHAT is its computational complexity. As it involves a grid search and the evaluation of multiple cross-correlations, computational cost can be prohibitive for certain applications. A great number of solutions to address this problem have been presented in literature, two of which we are presented as follows:

- Stochastic region contraction [22]: Consists of evaluating $P_{\mathrm{SRP}}(\mathbf{r})$ in a given volume, selecting a subset of the best solutions and restricting the volume to a region that contains those points. Doing this in an iterative way, the defined volume is shrunken to a sufficiently fine one (according to some criterion). Although this leads to a small number of evaluations of the SRP functional map, the nondeterministic nature of SRC can lead to poor results.
- Volumetric steered response power [23]: instead of a grid points search, a volumetric search is performed, which uses a coarser grid. The working volume is systematically decreased using a branch-and-bound paradigm, until a small enough volume is achieved and, finally, classical SRP can be applied.

3.3 MUSIC

The multiple signal classifier (MUSIC) [1] falls into the class of SSL solutions called subspace methods. The common feature of these methods is the exploitation of the eigenstructure of the signals spatial correlation matrices (SPCMs).

Starting from Equation (2.10), we can apply the Fourier transform to obtain

$$\vec{\mathbf{y}}(\omega) = \vec{\mathbf{x}}(\omega) + \vec{\mathbf{v}}(\omega) = S(\omega)\boldsymbol{\varsigma}(\omega, \mathbf{r}_s) + \vec{\mathbf{v}}(\omega), \tag{3.25}$$

where we have defined the steering vectors $\boldsymbol{\varsigma}(\omega, \mathbf{r_s}) = \left[e^{-\jmath\omega\tau_{11}(\mathbf{r}_s)} \quad \dots \quad e^{-\jmath\omega\tau_{M1}(\mathbf{r}_s)}\right]^T$. In order to keep notation more readable, we shall omit dependency on ω wherever necessary. We can calculate the narrowband SPCM $\mathbf{R_{yy}}(\omega)$, which is defined as

$$\begin{aligned}\mathbf{R_{yy}}(\omega) &\triangleq \mathbb{E}\left[\vec{\mathbf{y}}(\omega)\,\vec{\mathbf{y}}(\omega)^H\right] \\ &= \mathbf{R_{xx}}(\omega) + \mathbf{R_{vv}}(\omega) \\ &= \sigma_s^2\boldsymbol{\varsigma}(\mathbf{r}_s)\boldsymbol{\varsigma}(\mathbf{r}_s)^H + \sigma_v^2\mathbf{I},\end{aligned} \tag{3.26}$$

where we have defined the source and noise covariances as [1]

$$\begin{aligned}\sigma_s^2 &= \mathbb{E}[|S(\omega)|^2], \\ \sigma_v^2 &= \mathbb{E}[|V_1(\omega)|^2] = \dots = \mathbb{E}[|V_M(\omega)|^2],\end{aligned} \tag{3.27}$$

respectively.

Equation (3.26) writes the SPCM as a sum of a rank 1 Hermitian matrix $\mathbf{R_{xx}}$ and an identity (therefore full rank) matrix $\mathbf{R_{vv}}$. This leads to a very particular eigendecomposition [1], where the eigenvectors matrix $\mathbf{B}$ and their associated eigenvalues diagonal matrix $\mathbf{\Lambda}$ are given by

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_M \end{bmatrix}^T \tag{3.28}$$

$$\mathbf{\Lambda} = \mathrm{diag}[\lambda_s + \sigma_v^2, \sigma_v^2, \dots, \sigma_v^2]. \tag{3.29}$$

Combining Equation (3.28) with the eigenvalue equation, and with result (3.26) we find the remarkable result

$$\mathbf{b}_m^H \boldsymbol{\varsigma}(\mathbf{r}) \begin{cases} = 0 & \mathbf{r} = \mathbf{r}_s \\ \neq 0 & \text{elsewhere.} \end{cases} \tag{3.30}$$

The result found in (3.30) states that the eigenvectors associated with the eigenvalue σ_v^2, i.e. $\mathbf{b}_m$, $m \geq 2$, are orthogonal to the steering vector corresponding to the correct source direction $\mathbf{r}_s$. These eigenvectors are called noise eigenvectors, and the subspace they span is called the noise subspace. Also, since $\mathbf{B}$ must form a complete basis for the M dimensional space, $\mathbf{b}_1$ must have the same direction of $\boldsymbol{\varsigma}(\mathbf{r}_s)$. For this reason, it is called the signal eigenvector, and accordingly, it spans the signal subspace[2]. In order to take advantage of this orthogonality, we can define a power-like map

$$P_{\mathrm{MUSIC}}(\omega, \mathbf{r}) = \frac{1}{\sum_{m=2}^{M} |\mathbf{b}_m^H \boldsymbol{\varsigma}(\mathbf{r})|^2}, \tag{3.31}$$

and the associated source location is given by

$$\hat{\mathbf{r}}_s^{\mathrm{MUSIC}} = \arg_{\mathbf{r}} \max P_{\mathrm{MUSIC}}(\mathbf{r}). \tag{3.32}$$

Since $P_{\mathrm{MUSIC}}(\mathbf{r}_s) \to \infty$, the maps generated via the MUSIC method presents very well defined peaks. However, the main disadvantage is the necessity of choosing a good value for ω. Since the success of the method relies on $\lambda_s \gg \sigma_s$, frequency component selection is critical. If the signals we wish to localize are broadband, it can be challenging to select a particular frequency to perform the method. Furthermore, when dealing with non-stationary signals, such as speech or music, this selection must be made on a frame-by-frame basis, which can be troublesome. Tackling this problem is an ongoing topic in research [24, 25]. A broadband variation of MUSIC is proposed in [1], which uses the time SPCM. Albeit it solves the frequency esti-

[2]Since the noise fills the whole spatial correlation space, the more appropriate (and less commonly used) term would be signal and noise eigenvector/subspace.

mation problem, it has a high complexity and does not produce peaks as prominent as its narrowband counterpart. Also, while the MUSIC method generalizes well for the case of multiple sources (as long as the number of sources is smaller than the number of microphones), the broadband approach does not work in this scenario.

3.3.1 Other Subspace Methods

A few other approaches exist within the subspace decomposition framework. A couple of those methods are presented below, but a more extensive list of methods can be found in [26].

- Weighted Subspace Fitting [27]: This method was conceived within the multiple sources framework, and it explores the fact that the signal subspace vectors have the same direction as the steering vectors corresponding to the correct source locations. By jointly minimizing the projection of the signal subspace vectors on the orthogonal subspace of the steering vectors, it is possible to find estimates of the different source positions. A broadband version also exists [2].

- ROOT-MUSIC [2]: When the microphone array being used is a ULA, the steering vectors take the form of a polynomial vector $\boldsymbol{\varsigma}(\theta_s) = \left[\, 1 \;\; z \;\; z^2 \;\; \dots \;\; z^M \,\right]^T$, where $z = e^{\jmath\omega \cos\theta/d}$. This means that Equation (3.30) can be seen as a polynomial equation. Since it is expected that a solution for such equation exist in the unitary circle, given by the value of z corresponding to the correct source position, one can use this information to identify the correct source position with no need to perform a grid search in the localization space, leading to the so-called ROOT-MUSIC method.

3.4 Neural Networks for SSL

There is a common problem found in the methods mentioned above: although in different degrees, every method loses localization capabilities in scenarios with a high level of reverberation and/or a low SNR. Meanwhile, with the great developments of artificial neural networks (NN) research and their unquestionable success on audio signal processing tasks, such as speech recognition [28] and dereverberation [29], it has become apparent that they could become a viable direction for SSL research. As expected, a multitude of NN systems for SSL have been developed.

There are many different ways of interpreting Neural Networks. In this work, we will do so by showing that they can be seen as a generalization of the logistic regression machine [30]. After this description, we will present examples to illustrate how NN have been used for SSL.

3.4.1 Logistic Regression and Linear Models

Consider the task of, given an input vector $\mathbf{x} \in \mathbb{R}^M$, finding the probability that the binary target variable t associated with $\mathbf{x}$ is 1. It is also assumed that there is a dataset of L observations of $\mathbf{x}_l$, denoted by $\mathbf{X}$, and their associated target values t_l, denoted by $\mathbf{t}$. This is the so called supervised density estimation task, and there are many different statistical models capable of performing such task [31]. The logistic regression machine takes a linear approach by introducing a scalar quantity s called the activation, given by

$$s = \mathbf{w}^T\mathbf{x} + b, \tag{3.33}$$

where $\mathbf{w} \in \mathbb{R}^M$ is the weight vector, used to combine linearly all the input features (i.e. the different entries in the vector $\mathbf{x}$), and b is a scalar term, called the bias. This quantity is used to produce the model output, given by $f(s)$. For the density estimation problem, $f(s)$ is given by the logistic function, presented in Figure 3.1, which is defined as

$$f(s) = \sigma(s) = \frac{1}{1 + e^{-s}}. \tag{3.34}$$

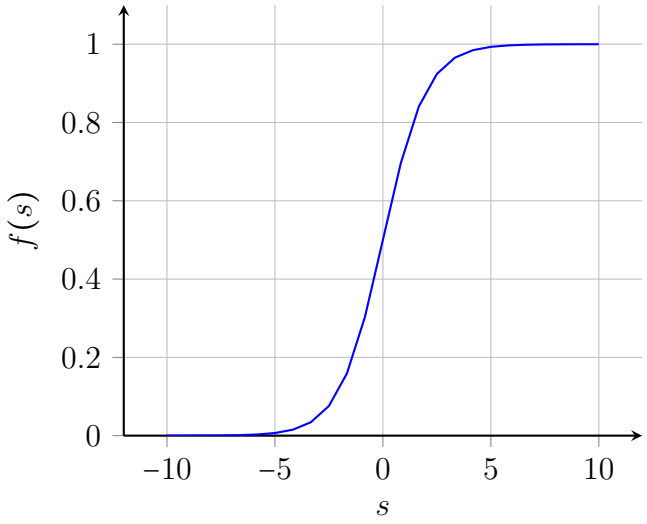


Figure 3.1: Logistic Function

In order to find the values of $\mathbf{w}$ and b, one usually resorts to gradient-based optimization techniques [30], such as gradient descent and its variants. These are applied to reduce some sort of error function which measures the difference between the logistic function output and the expected target for all the input samples in the dataset. For this task, the cross-entropy error function

$$L(\mathbf{X}, \mathbf{t}) = \frac{1}{L}\sum_{l=1}^{L} t_l \ln\left(\frac{1}{f(\mathbf{w}^T\mathbf{x} + b)}\right) + (1 - t_l)\ln\left(\frac{1}{1 - f(\mathbf{w}^T\mathbf{x} + b)}\right), \tag{3.35}$$

is usually minimized.

For tasks other than probability density estimation, the so-called activation function $f(s)$ can be replaced to make the output and the target compatible. For instance, if instead of identifying the probability of $t = 1$, one wishes to implement a detector, such that the output of the model $f(s) = 1$ if $t = 1$ and $f(s) = -1$ if $t = 0$, then it is possible to use the sign function, given by

$$f(s) = \text{sign}(s) = \begin{cases} 1 & s \geq 0 \\ -1 & \text{otherwise,} \end{cases} \tag{3.36}$$

which can be seen in the blue curve on Figure 3.2.

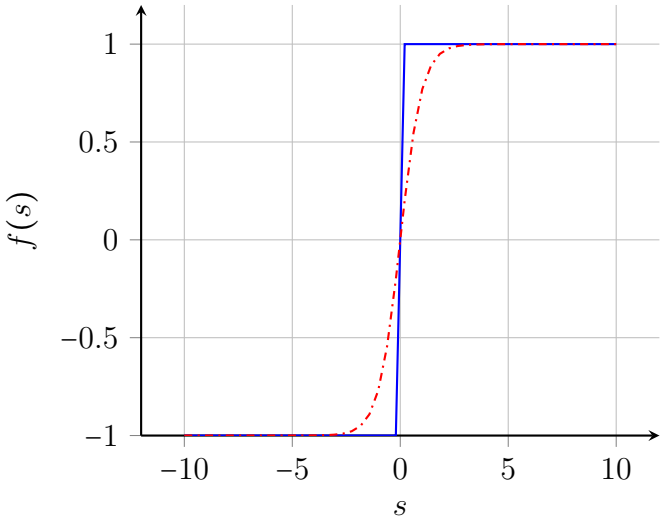


Figure 3.2: Sign function (Blue, continuous) and hyperbolic tangent (Red, dash-dotted).

This detector is commonly called the perceptron [30]. Although there are a few training algorithms available for the perceptron model, it is usually more common to initially treat the classification problem as a density estimation one. Thus, one can perform training on a logistic regression machine, and after acquiring $\mathbf{w}$ and b, use their values for the perceptron machine. This is usually justifiable, as Equation (3.36) can be rewritten as

$$\text{sign}(s) = \begin{cases} 1 & \sigma(s) \geq 0.5 \\ -1 & \text{otherwise.} \end{cases} \tag{3.37}$$

If the task of interest evolves around predicting a target that spans the real domain (a position, or an angle, in our case), one can use $f(s) = s$, arriving at the linear regression machine. These three models are often called linear models, and a common way to graphically illustrate them can be seen in Figure 3.3.

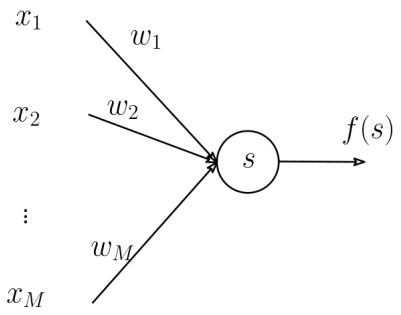


Figure 3.3: Graphical diagram of the linear model. Every component x_i of the input vector $\mathbf{x}$ is multiplied by the associated weight vector $\mathbf{w}$ component w_i. The bias term is omitted for simplicity.

3.4.2 Multi-layer perceptron

For many datasets, a linear model may be sub-optimal; in the task of classification, for instance, the perceptron can only correctly classify linearly separable data [30]. If the optimal decision boundary is a nonlinear manifold, a nonlinear model should be used. A simple and effective way to extend the perceptron for nonlinear classification tasks is to connect multiple perceptrons in a network as the one presented in Figure 3.4. The outputs of the perceptrons in layer $l-1$ are the inputs for the perceptrons in layer[3] l.

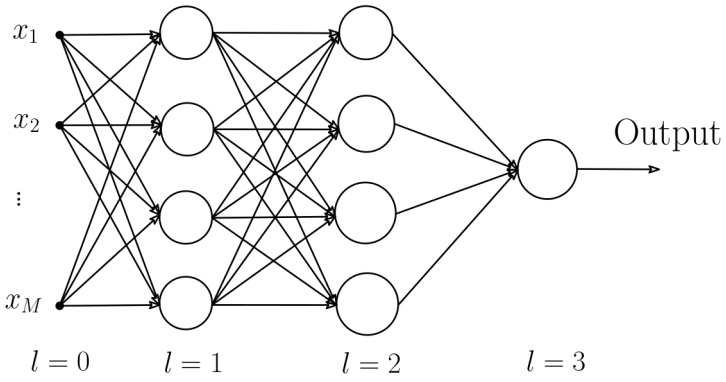


Figure 3.4: Graphical diagram of a neural network.

Activation Functions

Just like the linear model, the neural network is flexible in the sense of being possible to use it in many different tasks (classification, regression and density estimation). Moreover, since the only layer that has a task-specific activation function is the final one, it is possible to use different activations and activation functions in the other layers. Aside from the ones described above, the literature is filled with examples [7], a few of which we will describe in higher detail.

[3]In the machine learning community, the input features are assigned as layer $l = 0$ of the neural network.

The first one to be described is the hyperbolic tangent function (also known as tanh), which is given by

$$f(s) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{3.38}$$

and also illustrated on Figure 3.2, in the red curve. As can be seen in the figure, the tanh is a smooth version of the sign function. They are also often called the soft and hard threshold functions, respectively. The fact that the hyperbolic tangent is continuous and differentiable allows it to be used with gradient based optimization techniques. It is also easy to show that the tanh is a scaled and translated version of the logistic function. They are both part of a group of functions called sigmoids, due to their shape.

Whenever the problem in hands includes multiple classes, the standard approach is to use that same number of neurons in the output layer, and use softmax as the activation function. The softmax can be seen as a generalization of the logistic function for multiple classes. In the case of K classes, given the value of s_k as the activation of the k-th neuron, its output (and thus, the modelled probability of the input belonging to class k) is given by

$$\text{softmax}_k(s_1, \ldots, s_K) = \frac{e^{s_k}}{\sum_{l=1}^{K} e^{s_l}}. \tag{3.39}$$

The so called rectified linear unit (ReLU) is a piecewise linear function given by

$$f(s) = \begin{cases} s & s \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.40}$$

graphically illustrated by the solid curve in Figure 3.5, that has become a popular activation function in the machine learning community. Being almost linear, it enjoys the properties that make linear models simple to train via gradient based optimization [7]. Many different variations of the ReLU were tested on NN systems, and a few of them stood the test of time. One of those, which will be of great interest in this work, is the parametric rectified linear unit (PReLU). It is also a piecewise linear function, defined by the expression

$$f(s) = \begin{cases} s & s \geq 0 \\ as & \text{otherwise,} \end{cases} \tag{3.41}$$

where a is a learnable parameter that scales the response in the negative domain. Figure 3.5 illustrates the result of different values of a. The ReLU can be seen as a PReLU with constant $a = 0$.

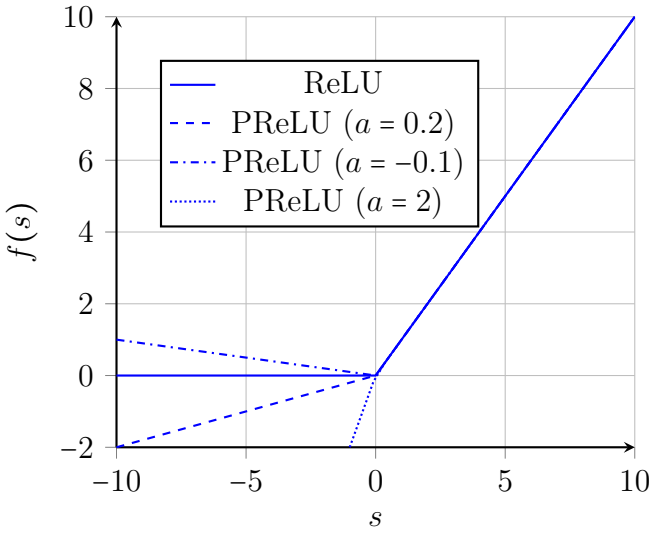


Figure 3.5: ReLU and different variations of the PReLU.

3.4.3 Input Features and Output Formats

When resorting to NN for a task, one has to answer the question of which input features should be used for learning. Although feeding raw data directly into a neural network is possible, better results can be attained by carefully designing the pre-processing and data transformation stage. For SSL, there has been plenty of different ways to transform multichannel audio signals into good input features. Classical choices of input features are the spectrogram [32–34] and the cross correlation map [35].

When it comes to SSL, another important debate is how to deal with the network output. One can deal with localization as a multi-class classification problem; in this case, the network output layer comprises K neurons, each one associated with a region in space. For example, if the goal is to identify the azimuth in a range from 0^o to 180^o, one can use 36 neurons and divide the θ space in 36 regions of 5^o each. Thus, the region associated to the neuron with the highest activation is taken as the source most probable position range. This output format has the advantage of easy extension to the multiple sources scenario, but has a limited resolution, defined by the number of classes. The other common approach is to deal with SSL as a regression problem. Depending on the localization type, one can use one, two or three neurons in the output layer, each representing a different target dimension. Although this approach seems more natural, it is sometimes deemed as more difficult to obtain a proper learning algorithm. A more detailed discussion on the output format topic can be found in [36, 37].

It should also be pointed out that hybrid approaches, although not so common, exist in the literature [38, 39]. They train NNs to provide outputs that in turn aid another localization method such as the ones previously mentioned. A complete

survey of deep learning methods for SSL that explores models with different input features and output formats can be found in [40].

3.4.4 Example: MUSIC-DNN

In order to motivate the proposed method, we are going to present an NN-based solution for SSL that also has its roots on the MUSIC method and was brought forward in [41]. In this approach, the eigenvectors of the narrowband SPCM, as presented in Section 3.3, are used as the input features of the network. Also, in the input layer of the network, an activation function called directional activation, which mimics the procedure of Equation (3.30), is defined as

$$f(\mathbf{x}, \mathbf{a}) = 1 - \frac{|\mathbf{a}^H \mathbf{x}|}{\|\mathbf{x}\|}, \tag{3.42}$$

where $\mathbf{a}$ is a learnable parameter that is expected to imitate the steering vectors. The NN also includes two layers with logistic activation functions to partially integrate all outputs from the directional layers, and a fully connected softmax output layer. The model adopts the classification framework for SSL. The success of [41] is an indication that using narrowband features and partially integrating them along the network can be very effective.

Chapter 4

Classical Solutions for Tracking

Whenever the source we are trying to track moves, it can be interesting to use previous information about its last estimated position (and even its velocity/acceleration) in order to estimate its current position. The tracking problem has a wide range of applications and different techniques. In this context, the Bayesian filtering framework [31, 42, 43] has been widely successful [44–46]. Also, in recent years, deep neural networks (DNN) have also been used to tackle acoustic source tracking using temporal context aware methodologies, such as temporal convolutional neural networks and recurrent neural networks. In this work, we are going to describe Bayesian filters in the two most common used flavors: Kalman filtering and particle filtering. Then, the deep learning (DL) solutions will be discussed and motivated.

4.1 Bayesian Filters

Consider the following setup: an unknown time varying quantity of interest $\mathbf{x}[n]$ is to be determined. For example, if we are interested in sound source tracking, a possible quantity of interest could be $\mathbf{x}[n] = \mathbf{r}_s[n]$. Time characteristics of the signal are described by an evolution model, given by

$$\mathbf{x}[n] = \mathbf{f}_n(\mathbf{x}[n-1]) + \mathbf{v}[n], \tag{4.1}$$

where $\mathbf{f}_n(\cdot)$ is a known function called the evolution model, and $\mathbf{v}[n]$ is comprised of noise (details on colouring and distribution are application specific). Instead of observing $\mathbf{x}$ directly, only a noisy measurement $\mathbf{z}$ is available, and their relationship is described by the measurement function $\mathbf{h}$, which is given by

$$\mathbf{z}[n] = \mathbf{h}_n(\mathbf{x}[n]) + \mathbf{w}[n], \tag{4.2}$$

where $\mathbf{w}$ is another random noise component. Depending on the literature, the pair $\mathbf{x}$ and $\mathbf{z}$ has a different naming convention. Here, we will refer to them as state and observation variables, respectively. The framework is based on two steps: a prediction step, where all the information available up to a time instant $n-1$ is used to predict the value of the state $\mathbf{x}[n]$; and then an update step, where we use information from the measurement $\mathbf{z}[n]$ to correct our estimate. As we are going to take a probabilistic approach, it will be useful to define a few quantities of interest. Namely, we will define the prior belief distribution as

$$\overline{\text{bel}(\mathbf{x}[n])} = p(\mathbf{x}[n] \,|\, \mathbf{z}[n-1], \ldots, \mathbf{z}[0]) = p(\mathbf{x}[n] \,|\, \mathbf{z}[n-1]), \tag{4.3}$$

and the belief distribution as

$$\text{bel}(\mathbf{x}[n]) = p(\mathbf{x}[n] \,|\, \mathbf{z}[n], \ldots, \mathbf{z}[0]) = p(\mathbf{x}[n] \,|\, \mathbf{z}[n]). \tag{4.4}$$

In both cases, the right-hand side equality was derived from the so called Markov assumption, which assumes that the states variables are complete, i.e if $\mathbf{z}[n-1]$ is known, knowledge from past observations does not convey any additional information to the model. The prior belief density represents the knowledge on the value of $\mathbf{x}[n]$ given all observations up to time instant $n-1$, and the belief distribution represents the knowledge on the value of $\mathbf{x}[n]$ given all observations up to time instant n (i.e., incorporating into the prior belief information coming from $\mathbf{z}[n]$). When an estimate of $\mathbf{x}[n]$ is necessary, one can find the value that maximizes $\text{bel}(\mathbf{x}[n])$, and take it as the most probable state value.

Since both Equations (4.1) and (4.2) have random components, we will describe the evolution and observation processes in terms of distributions $p(\mathbf{x}[n] \,|\, \mathbf{x}[n-1])$ and $p(\mathbf{z}[n] \,|\, \mathbf{x}[n])$ respectively, and we will assume that those distributions have some functional dependence on $\mathbf{f}_n(\mathbf{x}[n-1])$ and $\mathbf{h}_n(\mathbf{x}[n])$ (for instance, they can be the average of their respective distributions). With this in mind, and applying the theorem of total probability and the Bayes rule [42], we can derive the prediction and update steps of the Bayes filter

$$\overline{\text{bel}(\mathbf{x}[n])} = \int p(\mathbf{x}[n] \,|\, \mathbf{x}[n-1]) \, \text{bel}(\mathbf{x}[n-1]) \mathrm{d}\mathbf{x}[n-1] \tag{4.5}$$

$$\text{bel}(\mathbf{x}[n]) = \eta p(\mathbf{z}[n] \,|\, \mathbf{x}[n]) \overline{\text{bel}(\mathbf{x}[n])}, \tag{4.6}$$

where η is a normalization constant. The way these densities are modeled and calculated will lead to different approaches of Bayesian filters.

4.1.1 Kalman Filter

The Kalman filter, also commonly called the linear quadratic estimator (LQE), is a particular implementation of the Bayesian filter. It makes a few assumptions on the previously presented model, with the intention of making the problem tractable. The first assumption is that the evolution and observation functions are linear. This allows us to rewrite (4.1) and (4.2) as

$$\mathbf{x}[n] = \mathbf{A}_n\mathbf{x}[n-1] + \mathbf{v}[n], \tag{4.7}$$

$$\mathbf{z}[n] = \mathbf{C}_n\mathbf{x}[n] + \mathbf{w}[n]. \tag{4.8}$$

The second assumption made in the LQE framework is that both $\mathbf{v}[n]$ and $\mathbf{w}[n]$ are zero-mean white Gaussian noises, with correlation matrices $\mathbf{Q}_\mathbf{v}$ and $\mathbf{Q}_\mathbf{w}$, respectively. This leads to the conditional probabilities $\text{bel}(\mathbf{x}[n])$ and $\overline{\text{bel}(\mathbf{x}[n])}$ in the model being normal densities [31]. Since the Gaussian density can be fully characterized by its mean vector and correlation matrix, we only need to keep track of these entities along the filtering processes. In the Kalman filtering literature, the average and correlation matrix of $\text{bel}(\mathbf{x}[n])$ is called $\mathbf{x}_{n|n}$ and $\mathbf{P}_{n|n}$, and the parameters of $\overline{\text{bel}(\mathbf{x}[n])}$ are referred to as $\mathbf{x}_{n|n-1}$ and $\mathbf{P}_{n|n-1}$. Also, to simplify the algorithmic description, we will define the innovation quantity $\mathbf{y}_n = \mathbf{z}[n] - \mathbf{C}_n\mathbf{x}_{n|n-1}$, which is a measurement of the discrepancy between the predicted observation (based on the predicted state) and the actual observation. The innovation covariance matrix is given by $\mathbf{S}_n$.

The Kalman filter algorithm is described in Algorithm 1. A detailed mathematical derivation of it can be found in [42]. An important aspect of the LQE is the Kalman gain matrix $\mathbf{K}_n$. It is an indication of how much the information coming from $\mathbf{z}[n]$ is going to be incorporated into the current state estimate. It is simple to see that a zero matrix would keep $\text{bel}(\mathbf{x}[n]) = \overline{\text{bel}(\mathbf{x}[n])}$.

4.1.2 Particle Filter

The Kalman filter takes a parametric approach to implement the Bayes filter, as it models the belief and the prior belief as Gaussians. This would lead to problems if either the evolution model or the observation model were not linear equations. A few variations of the Kalman filter were proposed to deal with such problems, such as the extended Kalman filter [42], which linearizes both the evolution and observation models using Taylor series. Still, for highly non-Gaussian belief distributions (which can result from highly nonlinear models), this is still a poor aproximation.

The nonparametric approaches to the Bayes filter attempts to model the beliefs by decomposing their state space into a finite number of regions. One of such

Algorithm 1: Kalman Filter Algorithm

Input : $\mathbf{x}_{0|0}$, $\mathbf{P}_{0|0}$

$\mathbf{x}_{n|n} \longleftarrow \mathbf{x}_{0|0}$

$\mathbf{P}_{n|n} \longleftarrow \mathbf{P}_{0|0}$

for $n = 1, 2, \ldots$ **do**

- $\mathbf{x}_{n|n-1} = \mathbf{A}_n \mathbf{x}_{n-1|n-1}$; // Prediction Step (Mean)
- $\mathbf{P}_{n|n-1} = \mathbf{A}_n \mathbf{P}_{n-1|n-1} \mathbf{A}_n^T + \mathbf{Q}_\mathbf{v}$; // Prediction Step (Cov. Matrix)
- $\mathbf{y}_n = \mathbf{z}[n] - \mathbf{C}_n \mathbf{x}_{n|n-1}$; // Innovation
- $\mathbf{S}_n = \mathbf{C}_n \mathbf{P}_{n|n-1} \mathbf{C}_n^T + \mathbf{Q}_\mathbf{w}$; // Innovation Cov. Matrix
- $\mathbf{K}_n = \mathbf{P}_{n|n-1} \mathbf{C}_n \mathbf{S}_n^{-1}$; // Kalman Gain Matrix
- $\mathbf{x}_{n|n} = \mathbf{x}_{n|n-1} + \mathbf{K}_n \mathbf{y}_n$; // Update Step (Mean)
- $\mathbf{P}_{n|n} = (\mathbf{I} - \mathbf{K}_n \mathbf{C}_n) \mathbf{P}_{n|n-1}$; // Update Step (Cov. Matrix)

approaches is the particle filter (PF), which approximates their values using random samples drawn from the posterior distributions. At each time step, we keep the sets $\mathcal{X}_n$ and $\overline{\mathcal{X}_n}$, which store samples (also called particles) and weights, with the intent of modeling $\text{bel}(\mathbf{x}[n])$ and $\overline{\text{bel}(\mathbf{x}[n])}$ respectively. Each particle $\mathbf{x}^{[l]}[n]$ has an associated weight $w_n^{[l]}$, which represents the probability associated with that state value.

The PF is described in Algorithm 2. A complete mathematical description of it can be found in [42]. Sampling of $\mathbf{x}^{[l]}[n]$, as described in the step 5 of the algorithm, can be done using Equation (4.1), and the importance factor calculation is performed through Equation (4.2).

Algorithm 2: Particle Filter Algorithm

Input : $p(\mathbf{x}[0])$

$\mathcal{X}_n \longleftarrow \varnothing$

$\overline{\mathcal{X}_n} \longleftarrow \varnothing$

for $n = 1, 2, \ldots$ **do**

- **for** $l = 1, \ldots, L$ **do**
  - $\mathbf{x}^{[l]}[n] \sim p(\mathbf{x}[n] \,|\, \mathbf{x}^{[l]}[n-1])$; // Evolution Model
  - $w_n^{[l]} = p(\mathbf{z}[n] \,|\, \mathbf{x}^{[l]}[n])$; // Importance Factor
  - $\overline{\mathcal{X}_n} = \overline{\mathcal{X}_n} + (\mathbf{x}^{[l]}[n], w_n^{[l]})$;
- **for** $l = 1, \ldots, L$ **do**
  - Draw $\mathbf{x}^{[i]}[n]$ from $\overline{\mathcal{X}_n}$ with probability $w_n^{[i]}$; // Resampling
  - $\mathcal{X}_n = \mathcal{X}_n + (\mathbf{x}^{[i]}[n])$;

Particle Filters have been widely adopted in SST solutions. Suppose the quantity of interest is the Cartesian position of the sound source $\mathbf{x}[n] = \mathbf{r}_s[n]$, and that the observed signal is comprised of the TDoAs between each pair of microphones.

Although source motion can be characterized by a linear evolution model, such as the van-Keuk's mode [43], the measured quantity is usually taken to be the TDOA τ_{ij}, governed by Equation (2.6) and therefore a nonlinear function of the state variables. Thus, a nonparametric approach is well suited.

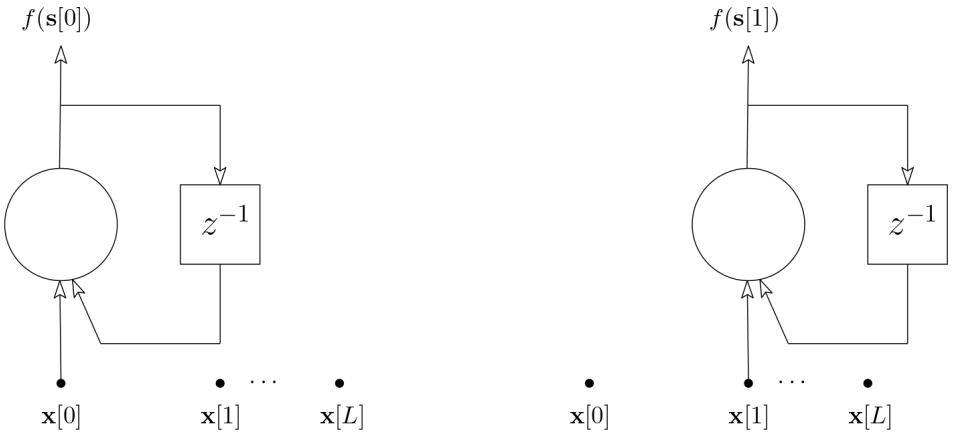
4.2 Neural Networks for Tracking

There are a few different ways to introduce temporal context into DNNs. Time varying targets usually require time varying input features in order to have a good tracking performance. A variety of solutions exist combining previously discussed input features with architectures that explore temporal correlations to perform their inference process.

4.2.1 Recurrent Neural Networks

Recurrent neural networks (RNN) [7] were developed specifically to deal with time series data. Many variations of RNN exist, but the main concept is the same across them. Assume a sequence of data points $\mathbf{x}[0], \mathbf{x}[1], \ldots, \mathbf{x}[L]$ such that for each of them an associated target $\mathbf{t}[0], \mathbf{t}[1], \ldots, \mathbf{t}[L]$ is provided. An RNN basic unit at time n uses as input not only $\mathbf{x}[n]$, but also a function of the previous activation, to compute the output. This allows the information from previous inputs to contribute to the current output.

The operation of a generic RNN is graphically depicted in Figure 4.1. To make the time evolution clearer, an unfolded version of the diagram is also provided in Figure 4.2, where the dashed units represent the previous activations, occurred at past time instants.



(a) Graphical diagram of an RNN at time 0. (b) Graphical diagram of an RNN at time 1.

Figure 4.1: Graphical diagrams of RNN at two subsequent time instants. The z^{-1} block denotes the unit time-delay operation.

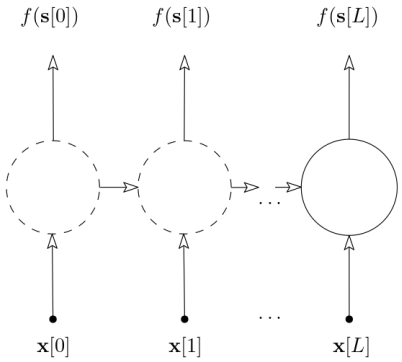


Figure 4.2: Graphical diagram of RNN unfolded.

Several variations of RNN units exist [47–50], each with different ways to incorporate information from past inputs. To give an example, we will present a simple implementation, using linear activations and the tanh activation function. The set of equations that characterize the forward pass in this RNN is given by

$$\mathbf{h}[n] = \tanh\left(\mathbf{W}\mathbf{x}[n] + \mathbf{U}\mathbf{h}[n-1] + \mathbf{b}\right) \tag{4.9}$$

$$\mathbf{s}[n] = \mathbf{V}\mathbf{h}[n] + \mathbf{c} \tag{4.10}$$

$$f(\mathbf{s}[n]) = \tanh\left(\mathbf{s}[n]\right), \tag{4.11}$$

where $\mathbf{W}$, $\mathbf{U}$, $\mathbf{b}$, $\mathbf{V}$ and $\mathbf{c}$ are the network parameters. The quantity $\mathbf{h}[n]$ is often called the state in the machine learning community[1], and its propagation through time is what allows the RNN to perform proper tracking. It is clear that the immediate predecessor state has a larger impact on the current estimate than older ones, which can be problematic for certain applications. Variations of the vanilla version presented in Equations (4.9) through (4.11) have been developed to provide the system with longer term memory [49].

4.2.2 Convolutional Neural Networks

Just like audio, image is a very particular type of signal. The identity of an image (or, in the machine learning framework, its class) is usually translation and scale invariant. It is also well known that closer pixels are more correlated than more distant ones. Furthermore, pictures usually present local features (such as edges and textures) that are spread along the whole scene.

Convolutional neural networks (CNN) [7, 31] were designed with the intent of analyzing images. They were specifically tailored to deal well with their particularities. As we shall see briefly, they use weight sharing to explore local features across the whole feature map and perform progressive sub-sampling.

[1] Although there is a similarity with the state variable defined previously, we will treat them differently and will not discuss this further.

Assume $\mathbf{x} \in \mathbb{R}^{L_1 \times L_2 \times 1}$. We will also define $\mathbf{a} \in \mathbb{R}^{L_1^l \times L_2^l \times D^l}$ as the output of the activation function at layer l. A common naming convention for $\mathbf{a}$ is feature map, and the dimensions L_1^l and L_2^l are called the map dimensions[2]. In a classical CNN, we have convolutional, pooling and activation layers, which are usually stacked in triplets along the network[3]. In classical CNN architectures, during the forward pass of the data through the network, there is a reduction of the feature map dimensions, as well as an increase in the number of maps D^l. This last dimension is often called the number of channels of the feature maps, which comes from the idea of color channels in an image. Although we are going to describe the details of the CNN procedure for image data and two dimensional feature maps, these concepts effortlessly generalize for three dimensional data, with so called three dimensional convolutions.

The parameters of a convolutional layer are a set of weight matrices $\mathbf{W}_k^l$ (also called filters or kernels) and a bias term. Both the feature maps and the filters are going to be presented as rectangular blocks, as seen in Figure 4.3.

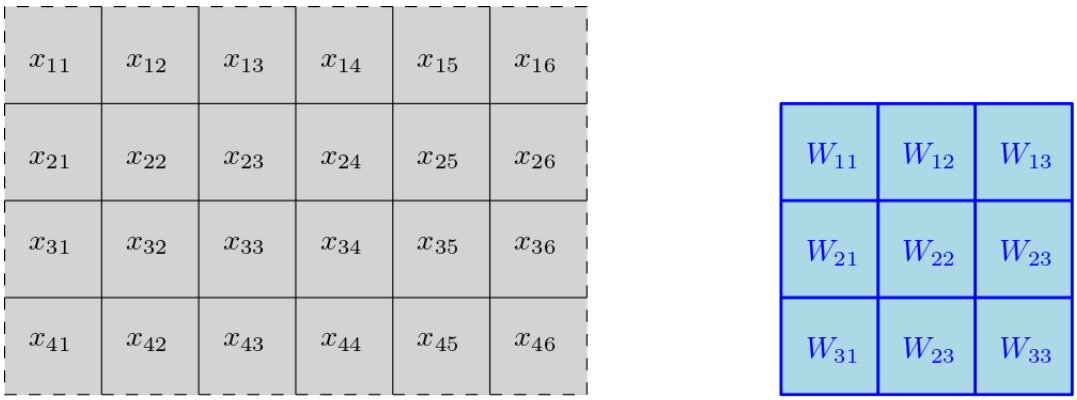


Figure 4.3: Graphical description of the feature maps (in grey) and weights (in blue) in a convolutional layer.

The main operation of the convolutional layer is the cross correlation between the filter and the feature map, which consists of applying a sliding point-wise product. In the graphical representation, this is classically represented by overlapping the filter with the image, as depicted in Figure 4.4. After performing the point-wise product between overlapped sections, the result is summed up, added to the bias, and stored in a map structure (which we will refer to as convolutional map), as depicted in Figure 4.4. The filter is shifted across the feature map, following the same procedure. This sliding process ensures that weights are shared along various regions of the feature map.

After the whole feature map is covered, the pooling operation is applied to the

[2]As in Section 3.4.2, we will refer to the input image as the feature map of layer $l = 0$

[3]In the literature the term "convolutional layer" often refers to the triplet convolutional / pooling / activation layer for brevity. In this work, we will refer to the triplet as a convolutional stack.

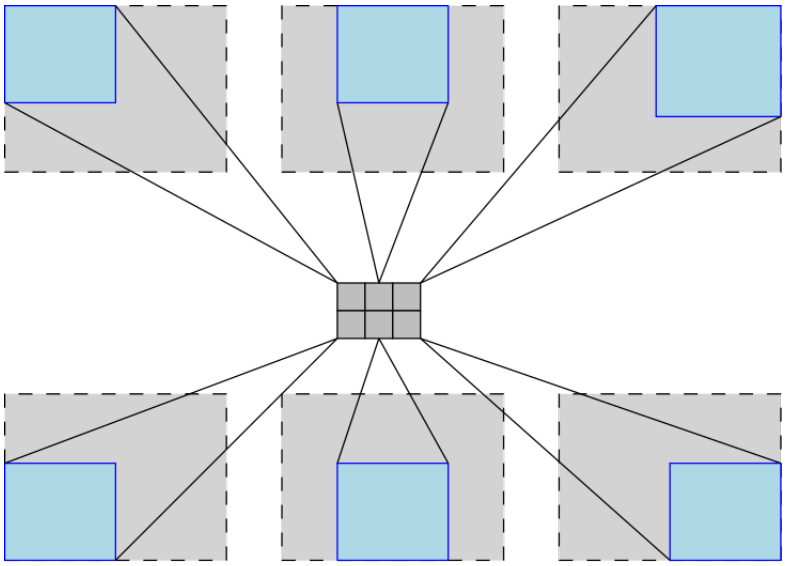


Figure 4.4: Graphical description of the convolutional procedure of the CNN. The grey block represents the feature map, and the blue block represents the filter.

outcome of each convolutional layer. The two most common pooling operations are max-pooling and average pooling. Pooling layers also use filters, but instead of calculating the point-wise product within the overlap of the filter and the convolutional map, they sub-sample the convolutional map, by either applying to the overlap between the map and the filter the maximum operation (in the case of max-pooling) or the average operation (in the case of average pooling). Afterwards, the result is passed along to the next feature map. This procedure is used to reduce even more the size of the feature maps, and is illustrated with an example in Figure 4.5.

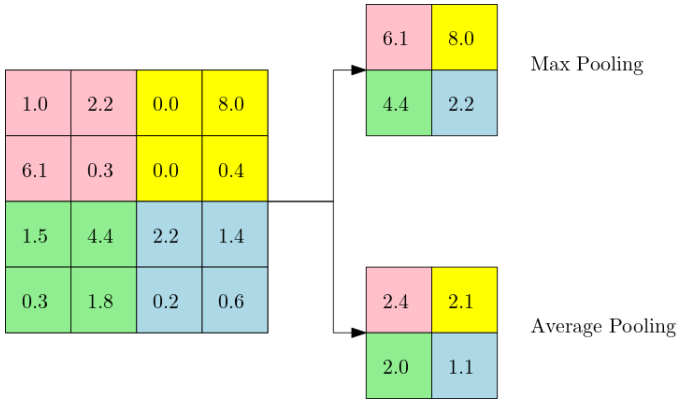


Figure 4.5: Graphical description of the pooling procedure of the CNN. Each color represents a section of the feature map that intersects with a different application of the pooling filter. The results of both types of pooling operations are illustrated.

Finally, an activation layer is applied, where an activation function such as the ReLU or PReLU, discussed in Section 3.4.2, is applied to every output of the previous pooling layer, finally arriving at another feature map.

The idea of the CNN is to use initial layers to identify simple features in an

image, such as the presence of edges in different directions, and in later layers of the network combine those low-level features into more complex ones, such as geometric shapes. This approach has an unquestionable success in the image processing field.

Padding

In the convolutional procedure, it is evident that the edges of a feature map will be less used in calculations than the center of the image. In some applications, this can lead to a bad performance. Also, the reduction in the feature map shapes along the network can also be problematic whenever spatial information is required. One of the many ways to tackle both these issues is to use padding. The idea of padding is to extend the feature map borders, as seen in Figure 4.6. The strategies used for this purpose are chosen based on the nature of input data and the supervised task at hand. The most common approaches are zero-padding (which fills the padded region with zeros), reflection padding (in which the padded region is symmetric with respect to the border) and circular padding (which "wraps" the feature map using the border values at the opposite end).

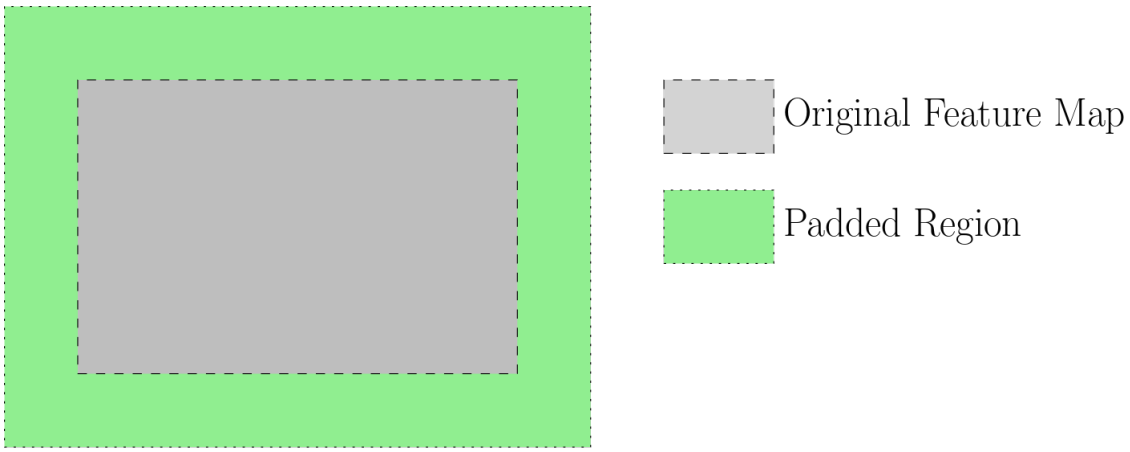


Figure 4.6: Graphical description of the padding procedure.

Stride

One aspect that can be defined in a convolutional layer is the stride used in the cross correlation process. A graphical description of the effect of changing its value can be observed in Figure 4.7. This parameter can be adjusted to control the feature map dimensions along the network.

Grouped convolutions

Another adjustable parameter is whether or not to use grouped convolutions. Instead of applying every kernel to every feature map, one can group a selection of

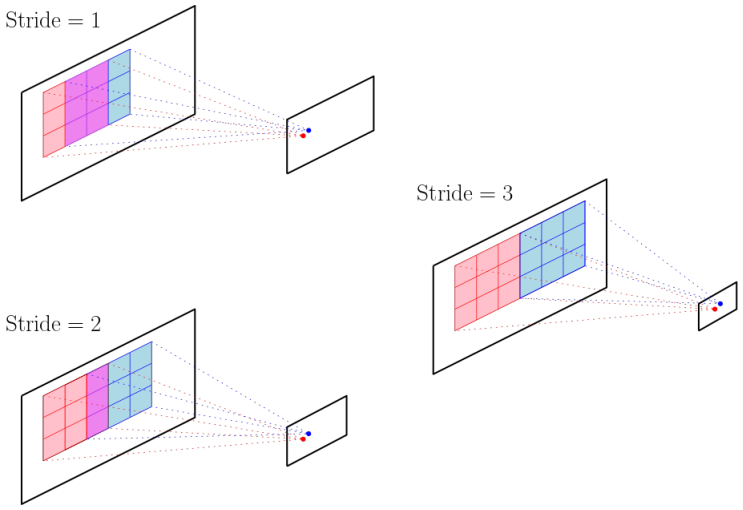


Figure 4.7: Graphical description of the stride parameter. When using a value of stride equal to one, the filter sliding process acts similar to the original cross-correlation procedure, as depicted on the top left image (where the red and blue blocks represent subsequent calculations, and the purple region represents their intersection). When using a higher value for the stride parameter, one can reduce the intersection, as seen in the bottom left figure, or even have no intersection, as in the right figure. Increasing the stride leads to a reduction in the next feature map size, as depicted on the figure.

kernels to operate on specific channels. If the number of output feature maps is held constant, grouped convolutions can reduce the number of filters, thus reducing the number of learnable parameters in the network. For instance, when selecting group = 2, it is equivalent to using two parallel convolutional layers, each seeing half of the feature maps and producing half of the output feature maps. This case is illustrated in Figure 4.8

Dilation

The last element of CNNs that we will focus on are the dilated convolutions. The dilation factor dictates the spacing between different points in the convolutional kernel. It can be applied to all dimensions of the kernel, or only to some directions. A visual description of how this parameter affects the convolutional procedure can be found in Figure 4.9. Although it seems uncanny to use a dilation factor other than 1 (which is the default for vanilla CNNs), large values of dilation are used in the temporal convolutional neural networks (TCNN) [51]. Whenever one of the dimensions of the feature map represents time, using a high dilation factor along that dimension allows the filters to observe information from time frames that are further apart, thus exploring long-term temporal correlations in the data. TCNNs can be modified using adequate padding to perform only causal computations, which would be the case in real-time applications such as tracking.

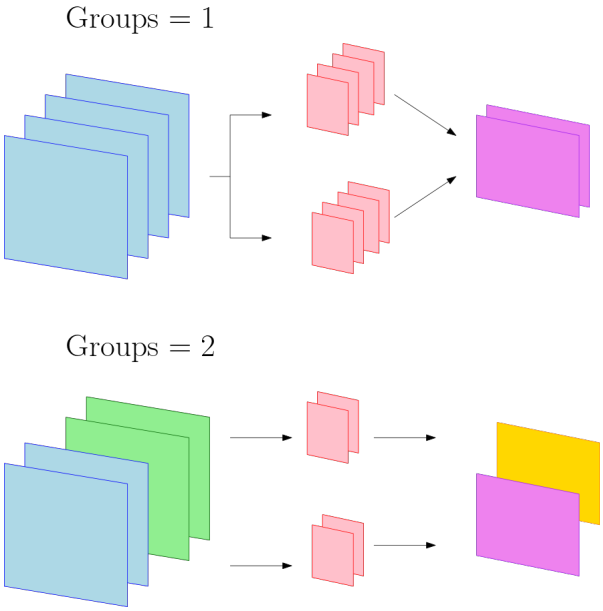


Figure 4.8: Graphical description of grouped convolutions. The top image exemplifies the scenario when there is no grouping (i.e. the groups parameter is set to one) and all of the convolutional kernels (in red) are applied to all of the input (in blue) channels, generating the output (in violet). Meanwhile, the bottom image illustrates the case when the groups parameter is set to two, when half the kernels are applied to half the input channels, generating half of the output channels.

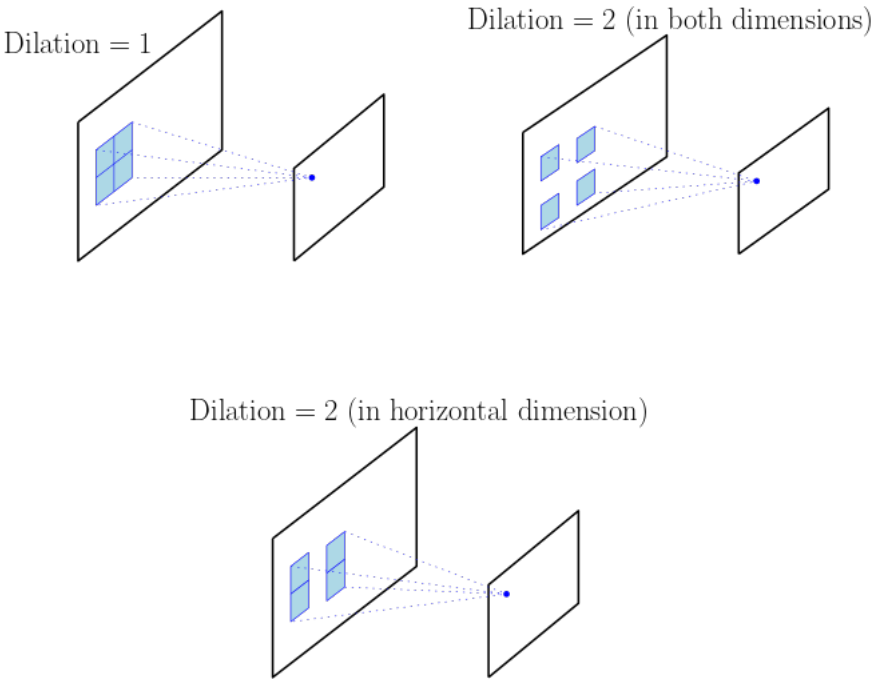


Figure 4.9: Graphical description of dilated convolutions.

Both TCNNs and RNN have been used successfully in SST literature. An extensive survey on different methods based on DNN can be found in [40].

4.2.3 Example: Cross3D

One of the most successful applications of DNN for the DoA tracking problem is the Cross3D model [52]. It combines the previously mentioned SRP-PHAT power maps, taken as input features, and a TCNN architecture that outputs a three dimensional vector. This vector is used to estimate the source position, and the Euclidean distance between this vector and the unitary vector that points to the correct source location is used as the loss function for training.

The pre-processing stage of Cross3D works as follows. Firstly, the system applies a 4096-sample Hanning window (with 1024 samples of overlap) to the microphone signals, dividing it into time frames. Then, a voice activity detector is used [4] to identify the silent frames. The SRP-PHAT power maps are then calculated and structured in a three dimensional tensor, with dimensions $R_\theta \times R_\phi \times T$, that represent the azimuth, the elevation and the time dimensions respectively. Also, the values of θ and ϕ that maximize each power map (which we will represent by θ^* and ϕ^*) are identified, originating two constant maps for each time frame. These constant maps (called max tensors) are gathered together with the previously calculated maps for the input of the network. In the case of silent frames, the power maps are replaced by constant zero blocks. This is illustrated in Figure 4.10

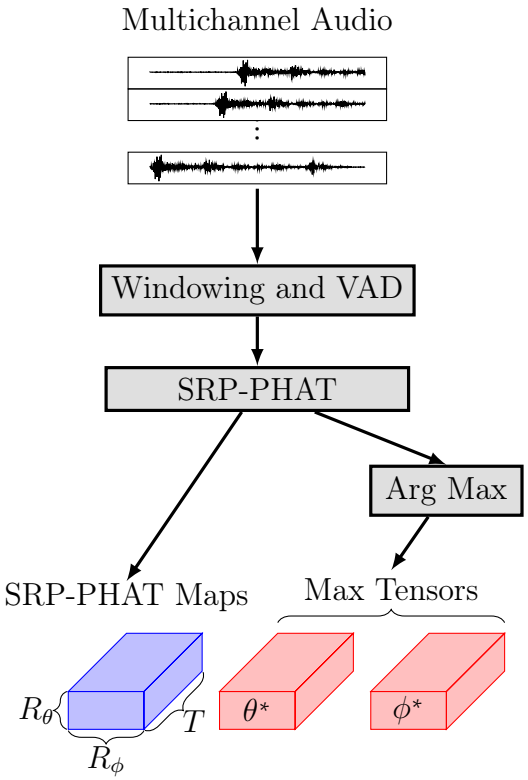


Figure 4.10: Graphical description of the preprocessing procedure in Cross3D.

[4] The system assumes a speech-like source.

As mentioned in the previous section, traditional CNNs take a subsampling approach by applying systematic pooling operations after each convolution. Although this is effective for image classification tasks, whenever we wish to find spatial information encoded in the feature maps, the pooling procedure may be inefficient [52]. To remedy this, after some initial three dimensional convolutional stacks, the architecture uses two "paths", each containing stacks that perform pooling on along one specific dimension of the feature map, keeping the other dimension fixed along the network. These paths are combined in later layers of the network, providing the final inferred DoA. This architecture is illustrated in Figure 4.11. The fact that three dimensional convolutions are used and that this crossed pooling procedure is used explains the name Cross3D.

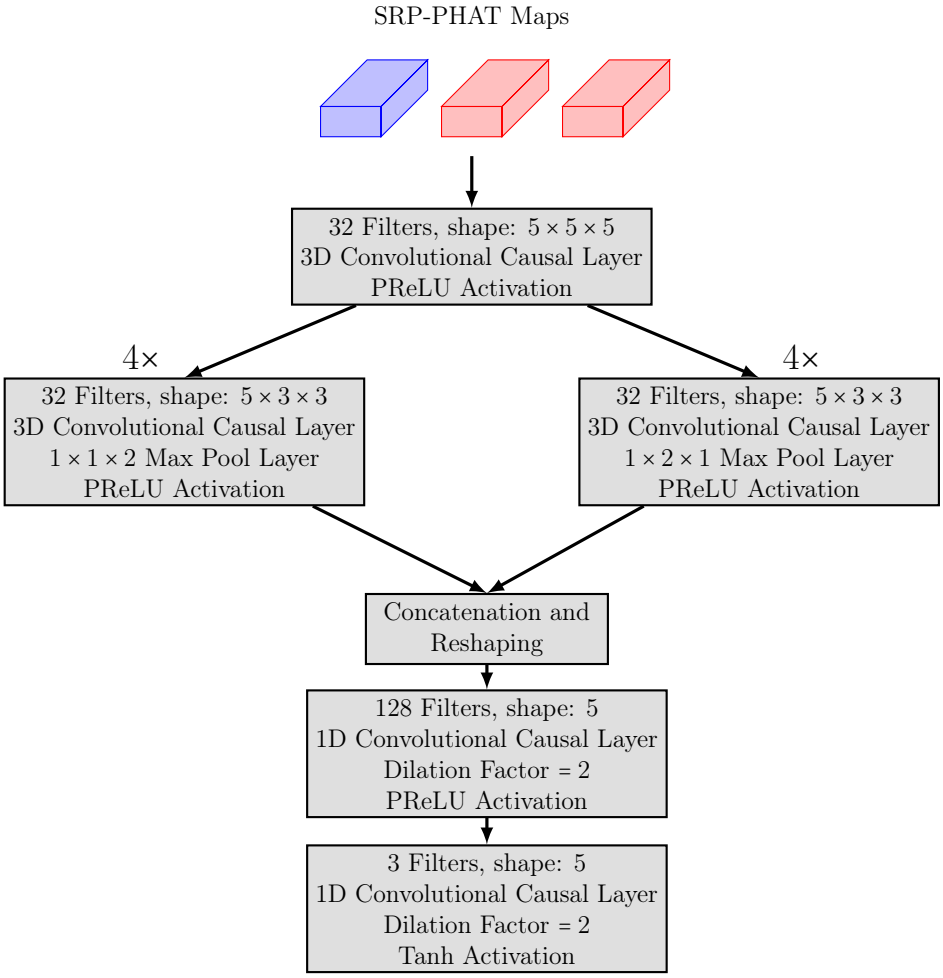


Figure 4.11: Graphical description of the neural network architecture in Cross3D.

The success of Cross3D is an indication that the combination of classical localization approaches with DNNs can perform better than either solution alone. The flexibility provided by the large number of parameters allows these approaches to overcome the difficulties that rise from adverse environments, such as low SNR and high reverberation.

Chapter 5

Proposed Solution: Spectral Cross3D

After presenting the theoretical background and examining a few solutions for SSL and SST, we can now introduce our proposed method, called the Spectral Cross3D (SC3D). In terms of proximity, our approach is highly inspired by the Cross3D solution [52], but with input features that resemble those of the MUSIC-DNN [41] solution.

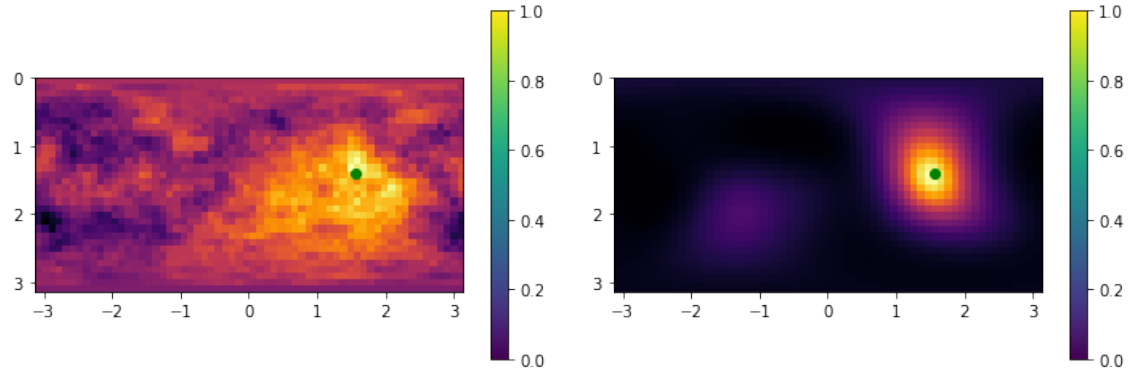
5.1 Preprocessing stage

One of the main points in the Cross3D solution is the aforementioned fact that it uses a well established solution for SSL, the SRP-PHAT, and enhances its capabilities by applying it to a TCNN. In our solution, we replaced the SRP-PHAT maps $\mathbf{P}_{\text{SRP-PHAT}}(\mathbf{r})$ with the MUSIC power-like maps $\mathbf{P}_{\text{MUSIC}}(\mathbf{r}, \omega)$. This preprocessing choice is similar to that used in MUSIC-DNN, but instead of delivering the raw eigenvectors of the narrowband SPCM, we perform the operations involving the steering vectors and the power-maps generation. To illustrate the difference between both types of map, we generated one sample from each technique, displayed on Figure 5.1.

For the initial approach, we chose to generate maps for all frequencies in the discrete spectrum. In later approaches, we included some frequency selection mechanism in the preprocessing chain. The performances attained with selected frequencies and with the full spectrum are compared.

5.1.1 MUSIC Power Map Calculations

The main issue with implementing the MUSIC method is how to compute matrix $\mathbf{R}_{\mathbf{yy}}(\omega)$. Equation (3.26) shows that it would be necessary to compute the expected value of $\vec{\mathbf{y}}(\omega)\,\vec{\mathbf{y}}(\omega)^H$. In mathematical terms, this would imply an infinite observation of all values of $\mathbf{y}$, which is impractical. A viable path is to rely on the ergodicity



(a) Map generated using the SRP-PHAT method.

(b) Map generated using the MUSIC method.

Figure 5.1: Maps generated using the SRP-PHAT and the MUSIC methods. In both cases, the green dot represents the actual source position.

approximation [53], and use a time average instead of the statistical mean. Although there are limitations on the effectiveness of this solution, it is widely adopted and shall be used here for this purpose.

Starting with the M microphone signals $y_1[n], \ldots, y_M[n]$ sampled at 16 kHz, we applied a rectangular 4096-point window, with 1024 samples of overlap (or, equivalently, a 3072-sample hop size), leading to T time frames. For each of them, a Hanning 256-point window with 32-point overlap is applied in order to perform a short-time Fourier transform (STFT). This leads to a tensor $\hat{\mathbf{Y}}$ with dimensions $M \times J \times F$, where J is the number of time windows[1] and F is the number of frequencies (in this case, $F = 128$). For a fixed frequency, the narrowband SPCM can be approximated using

$$\hat{\mathbf{R}}(\omega) = \frac{1}{M} \sum_{j=1}^{J} \hat{\mathbf{Y}}(j,\omega) \hat{\mathbf{Y}}^{H}(j,\omega). \tag{5.1}$$

From this information, it is possible to generate the MUSIC maps through the procedures described in Section 3.3. Subsequently, it is possible to identify the coordinates of the maxima from each map, which are stored for later use as side information in later layers. Finally, the maps are concatenated into a four dimensional tensor $\mathbf{Y}$ with shape $R_\theta \times R_\phi \times T \times F$, where R_θ and R_ϕ are, respectively, the number of grid-points in the azimuth and elevation dimensions. This pre-processing chain is illustrated in Figure 5.2, where green blocks highlight modifications with respect to Cross3D.

[1] It is important to notice that we are applying windowing to an already windowed signal.

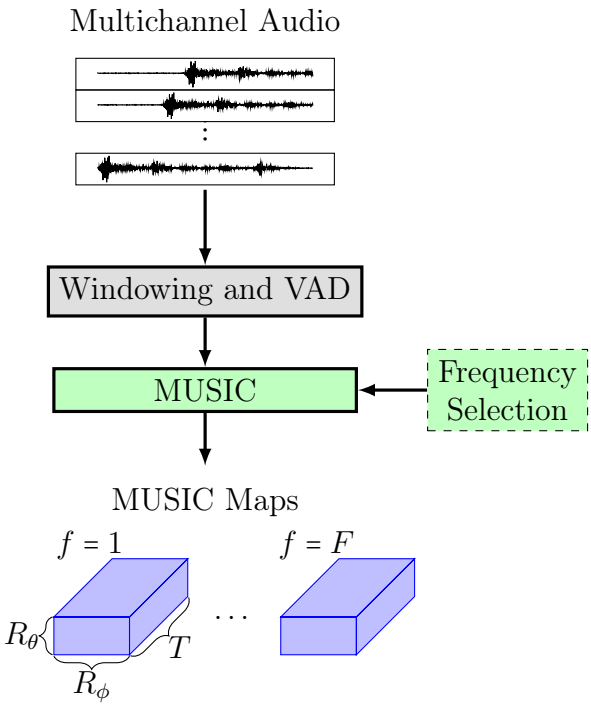


Figure 5.2: Graphical description of the pre-processing procedure in Spectral Cross3D.

5.1.2 Frequency Selection

Since for the MUSIC method to work properly it is important that $\lambda_s \gg \sigma_s$, many frequency maps may contain no information for localization. Also, the spatial aliasing phenomenon [1] states that the spatial correlation can present artificial peaks for frequencies above a given threshold that depends on the array geometry. These facts suggest that providing MUSIC maps for all frequencies might be sub-optimal. Even though there is an expectation that the neural architecture may learn how to properly identify usable maps and to combine them, reducing unnecessary data features is always an advisable strategy [30]. With this in mind, another step in the pre-processing chain (to be introduced later in this work) is the selection of the best set of frequencies to be delivered to the network. Two mechanisms were devised to this end, namely a spectral based selection and a map based selection.

Spectral Based Selection

The first approach consists of using the information in tensor $\hat{\mathbf{Y}}$ to estimate the signal power spectral density (PSD) [53, 54], and then use it to select the best candidate frequencies to feed the inference system. Given the non-stationarity of speech signals, it is apparent that this procedure must be performed on a frame-by-frame basis.

For the selection part, we considered the possibility of selecting F^* frequencies

that maximize the PSD, with the option of performing this selection by first dividing the spectrum in L equally spaced sub-bands and then selecting $\frac{F^*}{L}$ frequencies from each sub-band.

The main advantage of using this scheme is that the calculation of the power-like maps, which is a computationally heavy process, will be performed only F^* times, instead of F. If we have $F^* \ll F$, the computational time savings would be significant. However, this selection criterion has the disadvantage of performing the choice without actually seeing the maps. It is implicitly assumed that the best maps to provide useful information to the network are those with the maximum PSDs, but there is no guarantee that they will actually convey useful information.

Map Based Selection

The second approach consists of performing the selection after calculating the maps, and using some criterion to evaluate the quality of the map prior to selecting it. As the main point of the MUSIC method is to provide a peak at the correct position of the sound source, we adopted a sparsity criterion such that only maps with the smallest l_1-norm are presented to the network, i.e.,

$$\|\mathbf{Y}(t,\omega)\|_1 = \max_{\theta}\left(\sum_{\phi}|\mathbf{Y}(\theta,\phi,t,\omega)|\right). \tag{5.2}$$

An option to perform this selection after dividing the spectrum in sub-bands, as in the frequency selection criterion, was also provided. Just like the spectral selection, this procedure is done on a frame-by-frame basis.

The main advantage of this method is the fact that the maps are observed and taken into account in order to perform the selection. While the PSD can indicate frequencies that could potentially generate high quality maps, directly observing the maps can lead to a much better performance. The drawback, however, is that this method requires the calculation of all input maps before performing the selection. Thus, not only there is no computational savings in the preprocessing stage, but its complexity is increased by the l_1 norm calculation for each map.

5.2 Neural Network Architecture

In terms of the DNN used, we decided to follow an approach very close to the one used in Cross3D. This is because one of the goals in this work was to evaluate whether or not narrowband MUSIC maps could lead to a better performance than SRP-PHAT maps.

As the changes in the preprocessing stage lead to input features with different

dimensions, it was necessary to perform minor modifications to the DNN architecture to accommodate them. Also, it is expected that input data with higher dimensions require more parameters to perform the inference. Thus, it was decided to add to the network two convolutional stacks, using grouped convolutions, and keep the remaining layers of the network as they were. These convolutions were designed in a way not to look into past feature maps when generating their output, thus not increasing the temporal receptive field.

Also, we decided to incorporate information from the map maxima into the fully connected layers at the end of the network, instead of feeding them as input maps. All changes made in the architecture of Cross3D are indicated as green blocks in Figure 5.3.

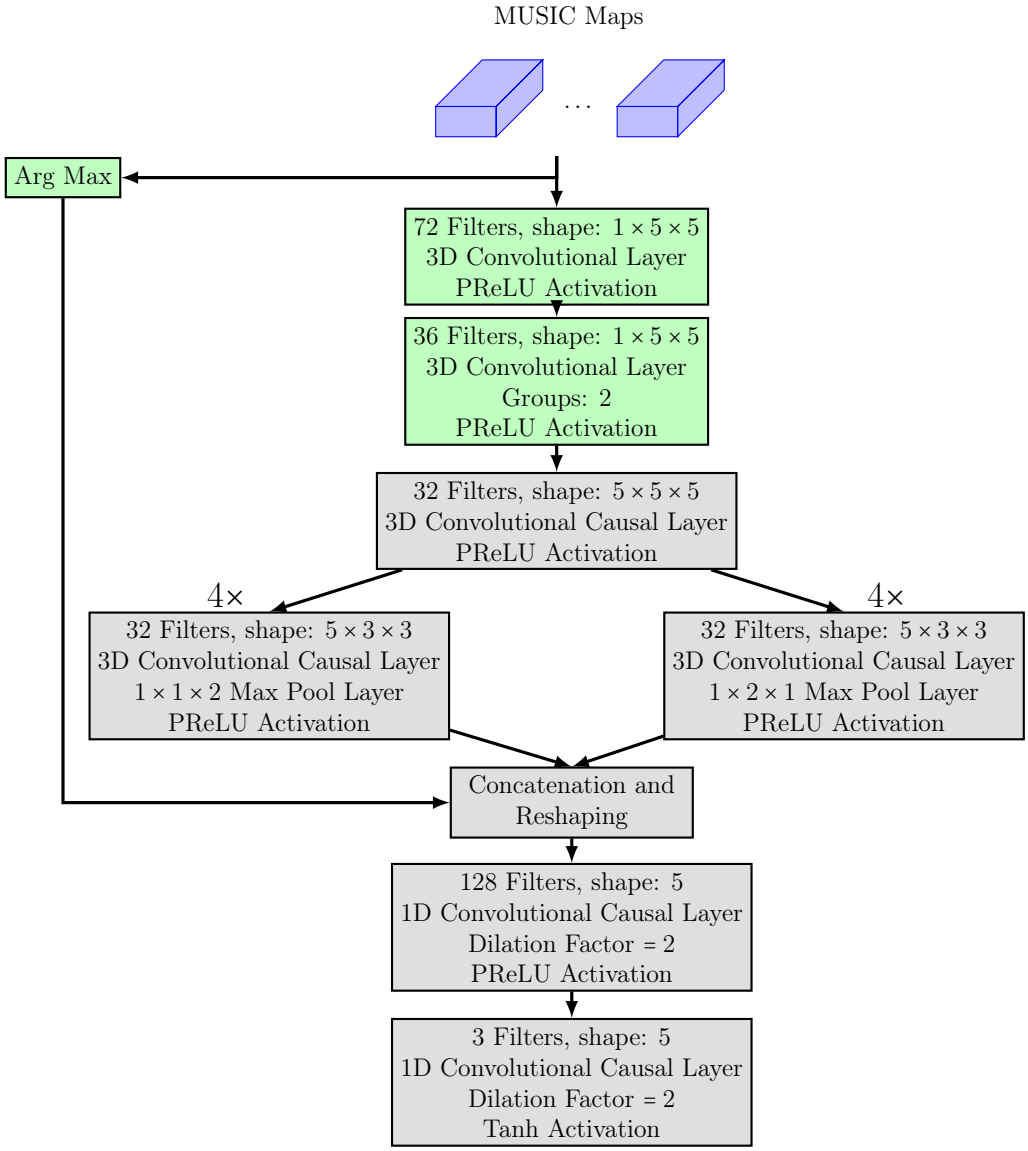


Figure 5.3: Graphical description of the neural network architecture in Spectral Cross3D.

Chapter 6

Experiments and Results

In this chapter, we motivate the experiments we performed and analyze their results. The scheme of the experiments is as follows. First, a speech signal is selected from a corpus dataset. It is subjected to a random trajectory with the aid of an acoustic simulator that generates the corresponding signals that would reach each microphone, taking into account noise and reverberation effects. An epoch of the training procedure of the network consists of a whole pass over the speech corpus, with one random trajectory generated per signal. A detailed explanation of the data generation and network training procedure, as well as a discussion of the attained results is given in the following sections.

6.1 Data Generation

6.1.1 Microphone Array

To perform our simulations, we selected a 12-microphone array designed for an NAO robot head [55]. Since we wished to perform simulations concurrently with the network training, we decided not to include scattering effects of the robot head within the simulation procedure. Since this array is not planar (i.e there is no two dimensional plane that contains all the microphones), we were able to perform elevation tracking from 0^o to 180^o.

6.1.2 Corpus

The speech signals selected were the ones in the LibriSpeech corpus [56], which is a compilation of audio-book readings, sampled at 16 kHz. This dataset has separate splits, called train-clean, dev-clean and test-clean. The first one was used in the network training process. The second split was used in the regularization and model selection stages. Finally, the third split was selected for a final evaluation and

comparison between the original Cross3D solution and our approach.

6.1.3 Trajectory Generation

To generate diverse trajectories, the same procedure used in [52] was adopted, which can be briefly described as follows. An initial $\mathbf{r}_0$ and a final $\mathbf{r}_L^*$ are chosen at random (within the limits of the simulated room). A linear trajectory is interpolated connecting the two points. Two vectors containing three random frequencies $\boldsymbol{\omega}$ and three random amplitudes $\mathbf{a}$ are sampled, and the linear trajectory is modified to give

$$\mathbf{r}_i = \mathbf{r}_0 + \frac{i}{L-1}(\mathbf{r}_L^* - \mathbf{r}_0) + \mathbf{A} \odot \sin(\boldsymbol{\omega} i), \tag{6.1}$$

where we have denoted as $\odot$ the point-wise product performed in the second term of the left-hand side. With a probability of 25%, the trajectory is ignored, and the simulated source is taken to be stationary at $\mathbf{r}_0$.

6.1.4 Simulator

Our simulated procedure consists on the following steps. A random room dimension is selected, within the range described in Table 6.1. Then, a random value for the reverberation time parameter is chosen uniformly in the range of 0.2 s to 1.3 s. Formally, the so-called RT60 measures how long in seconds the power at a given receiver takes to decay by 60 dB after a constant power source inside the room suddenly goes silent; it is usual to assign to a room an average RT60 without defining source and microphone positions. A value of signal-to-noise ratio (SNR) is also chosen, in a range that varies depending on the training stage of the network. While it starts at a fixed value of 40 dB, after 40 epochs it is reduced to a uniform random value between 5 dB to 30 dB. The array position inside the room is also randomly selected (guaranteeing a minimal separation from the walls of the room). Specifically, the values of x and y coordinates were chosen to be within the range of 10% and 90% of their maximum value, and the value of z is chosen in the lower half of the room.

Dimension	Min	Max
x	3 m	10 m
y	3 m	8 m
z	2.5 m	6 m

Table 6.1: Specification of the simulated rooms.

The simulator we used is the so-called gpuRIR simulator [57], which uses a GPU implementation of the image source method [58, 59] (ISM). Although there are

methods that perform more reliable simulations [60, 61], the gpuRIR is fast enough to be used on-line with the network training, allowing the system to be exposed to various scenarios during the training stage. This allows us to avoid overfitting to specific situations.

6.1.5 Voice Activity Detector

We used the WebRTC voice activity detection (VAD) system [62]. As we are training our system to work with speech signals, it makes sense to employ a system capable of identifying voice activity. For more general SSL and SST, a general sound activity detector could be trained in parallel with our system.

6.2 Neural Network Training

We performed the training of Spectral Cross3D (C3D) with a few variations of the preprocessing stage, as described in Section 5.1.2. Alongside the two previously described mechanisms, we also implemented an option to generate maps only up to a certain frequency. This was motivated by the fact that the spatial aliasing phenomenon [1] occurs at high frequencies. The trained systems were

1. SC3D using the full spectrum (referred to as SpC3D_1);
2. SC3D using the 32 highest values of the PSD (referred to as SpC3D_2);
3. SC3D using the 4 highest values in each of the 8 subbands the PSD has been split into (referred to as SpC3D_3);
4. SC3D using 32 frequencies centered around the spectral centroid (referred to as SpC3D_4);
5. SC3D using the 4 maps with the lowest l_1 norm in each of the 8 subbands the PSD has been split into (referred to as SpC3D_5); and
6. SC3D using frequencies lower than 4 kHz (referred to as SpC3D_6).

As the number of maps being fed to the network varies with the preprocessing mechanism, the number of channels in some layers of the network must be modified to accommodate these changes. The number of parameters for each system, including Cross3D, is shown in Table 6.2.

Table 6.2: Number of parameters in Cross3D and Spectral Cross3D architectures.

Model	# of parameters
Cross3D	5,626,148
SpC3D_1	6,591,188
SpC3D_2, SpC3D_3, SpC3D_4, SpC3D_5	6,161,108
SpC3D_6	6,304,468

6.2.1 Loss Function

We took as the loss function the Cartesian distance between the generated vector and the vector that points to the direction of the correct source position. This error is calculated and averaged over time, leading to a mean squared error loss function, which is minimized in the training stage.

Another useful function used in this work is the root mean squared angular error (RMSAE), which uses the great-circle distance [63], and calculates its root mean squared [1] (RMS) value across different time frames.

6.2.2 Optimizer

The network optimizer selected for the network parameters updating was the Adam optimizer [64], with coefficients β_1 and β_2 at their default values. It performs adaptive moments estimation to efficiently attain stochastic gradient-based loss minimization. A step scheduling strategy was also used for the algorithms' learning rate parameter γ, with a step size of 10 epochs and a decay factor of 0.8. The initial value of γ was set to 0.0003, and every 10 epochs the value was updated by

$$\gamma_{t+10} = 0.8\gamma_t, \tag{6.2}$$

where γ_{t+10} is the new value, and γ_t was the previous value. A mini batch size of 25 samples was implemented, i.e. the training parameters were updated after 25 samples had been observed.

6.2.3 Early Stopping

We also implemented the early stopping regularization technique [30]. It consists of stopping the training of the neural network when there is little or no increase in the accuracy of the system after a determined amount of epochs (called the patience factor), usually evaluated in another dataset. The selected patience factor was of 15

[1] $x_{\text{rms}} = \sqrt{\int_{-T}^{T} x^2(t)\mathrm{d}t}$

epochs, and the minimal relative decrease in the RMSAE necessary was of 0.1%. The dataset used in this evaluation was the dev-clean split of the LibriSpeech database.

6.3 Results

In order to properly compare the results attained in this work with those of [52], it was necessary to retrain the original Cross 3D architecture using the scheme presented in Section 6.2. This action had the goal of removing any bias unintentionally created by the training procedures when comparing the different models, making it possible to better grasp if the pre-processing stage was responsible for any increase in performance. Also, it is desired to select amongst all different preprocessing strategies the best suited to our system. Thus, we must undergo a two folded stage of model selection: one for comparing the original Cross3D system (which shall be referred to as C3D) with the retrained version (referred to as C3D*), and one for selecting each of the SpC3D preprocessing chain mechanisms presented previously.

6.3.1 Training time

All the systems mentioned above were trained on the same machine, described in Section 1.2. All the systems stopped their training at approximately 180 epochs. For the C3D* system, the training took two days. All the different versions of SpC3D took four days to train.

6.3.2 LibriSpeech

For the model comparison, we used the dev-clean split of LibriSpeech. This was the same split used for the early stopping criterion, but with different trajectories than those used in the training stage. A final evaluation is also performed, with the selected models from both architectures, in the test-clean split with another set of trajectories.

6.3.3 Model Selection

Cross3D

The results achieved with the original and the retrained Cross3D are displayed in Figure 6.1, for the root mean squared angular error (RMSAE) metric. As it can be seen, for every environmental setting, the retrained version attains a lower RMSAE value (on average, there is a reduction of 4.8%). Thus, comparing C3D* with our solution should allow for a fair assessment of the quality of the proposed method.

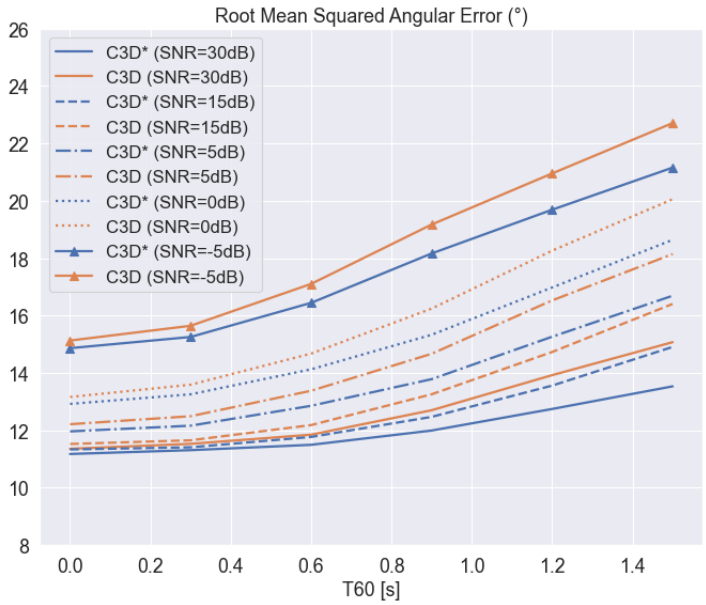


Figure 6.1: Comparison of the RMSAE attained by the original Cross3D (orange) and its retrained version (blue), for different values of SNR (different trace styles) and different reverberation times (x-axis).

Spectral Cross3D

The comparison of all different strategies applied to Spectral Cross3D can be seen in Figures 6.2, 6.3, 6.4, 6.5 and 6.6. The SpC3D_1, i.e, the full spectrum applied directly to the network, was clearly superior to all approaches in terms of the RMSAE, as it had the smallest RMSAE in every environmental condition. Meanwhile, the SpC3D_4 (which selects 32 frequencies around the spectral centroid) and the SpC3D_5 (which selects the 4 maps with the lowest l_1 norm in each spectral subband) had the highest RMSAE values in almost every environmental condition. The other approaches had varying levels of performance, none above that attained with SpC3D_1.

6.3.4 Final Evaluation

We performed a final evaluation of the two selected systems C3D* and SpC3D_1 using the test-clean split of LibriSpeech, and with newly generated trajectories. An example of how the two systems track an acoustic scene from the dataset can be seen in Figure 6.7.

The results attained are displayed in Figure 6.8. As can be seen, SpC3D_1 is superior in 25 scenarios (against 5 for C3D*). On average, SpC3D_1 provided a relative reduction of RMSAE around 9.1%.

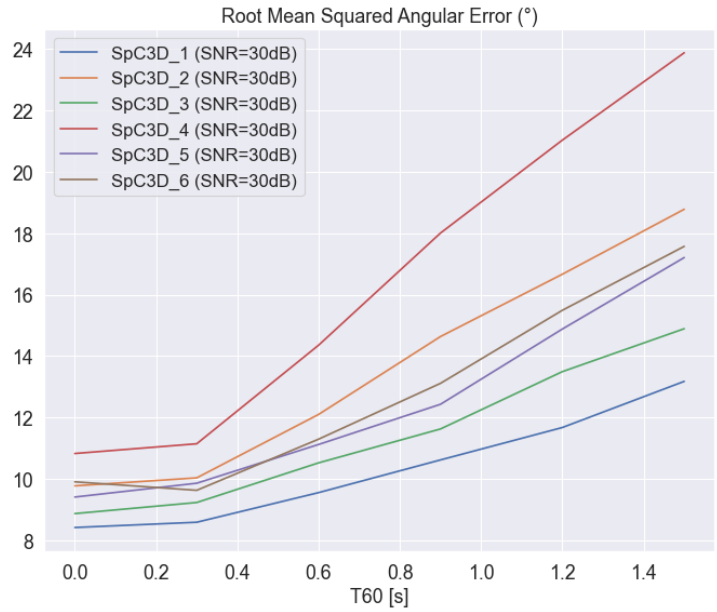


Figure 6.2: Comparison of the RMSAE attained with the different preprocessing chains of Spectral Cross3D, at an SNR of 30 dB — names specified in Section 6.2. The x-axis represents different RT60 values.

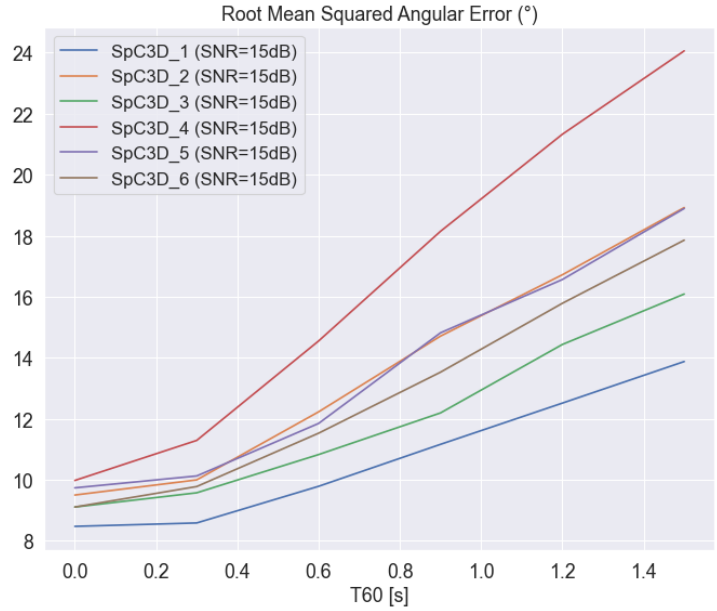


Figure 6.3: Comparison of the RMSAE attained with the different preprocessing chains of Spectral Cross3D, at an SNR of 15 dB — names specified in Section 6.2. The x-axis represents different RT60 values.

6.3.5 LOCATA

The acoustic source localization and tracking (LOCATA) Challenge [65] took place in 2018, and included a small set of audio signals for competitors to evaluate their SSL and SST systems. Although the small number of samples implies a high variance in the error evaluation, we found it useful to compare our system with the baseline

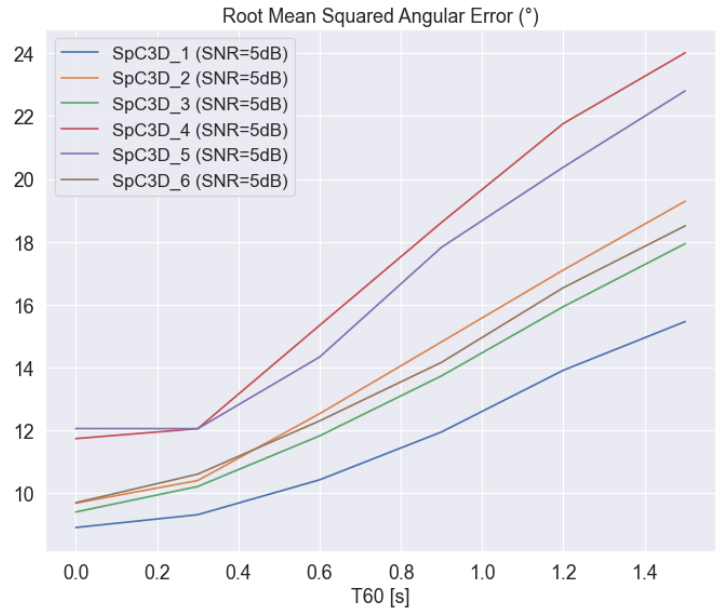


Figure 6.4: Comparison of the RMSAE attained with the different preprocessing chains of Spectral Cross3D, at an SNR of 5 dB — names specified in Section 6.2. The x-axis represents different RT60 values.

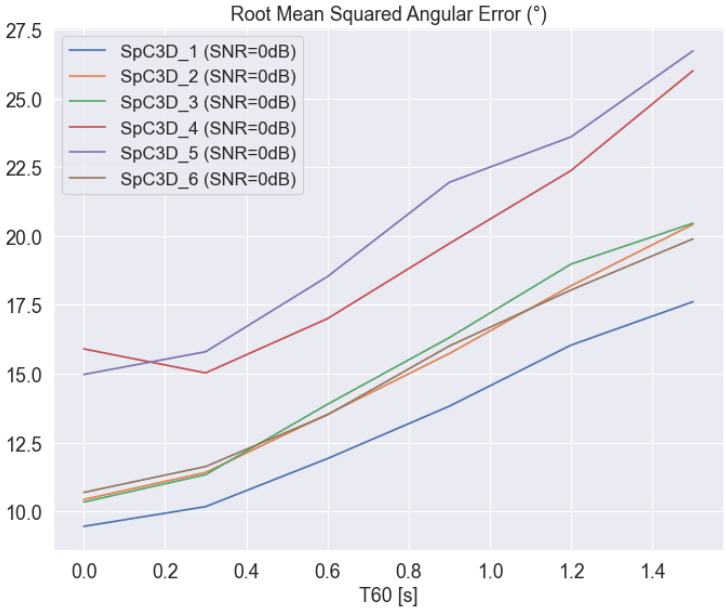


Figure 6.5: Comparison of the RMSAE attained with the different preprocessing chains of Spectral Cross3D, at an SNR of 0 dB — names specified in Section 6.2. The x-axis represents different RT60 values.

presented for the challenge and thus evaluate its viability.

The Challenge was comprised of 6 tasks, of which only three (Tasks 1,3 and 5) were single-source tasks. The description of each task is given as follows. Task 1 had static loudspeakers and a static microphone array. Task 3 consisted of tracking moving speakers with a stationary microphone array. Finally, task 5 presented

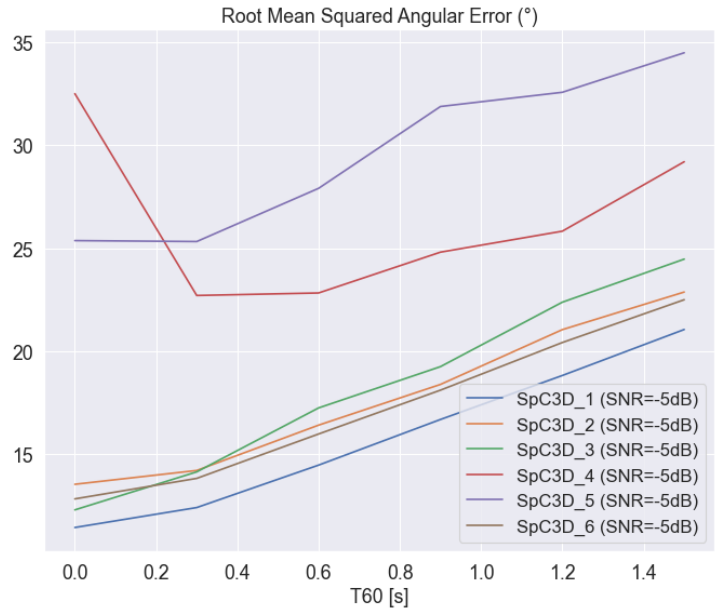


Figure 6.6: Comparison of the RMSAE attained with the different preprocessing chains of Spectral Cross3D, at an SNR of −5 dB — names specified in Section 6.2. The x-axis represents different RT60 values.

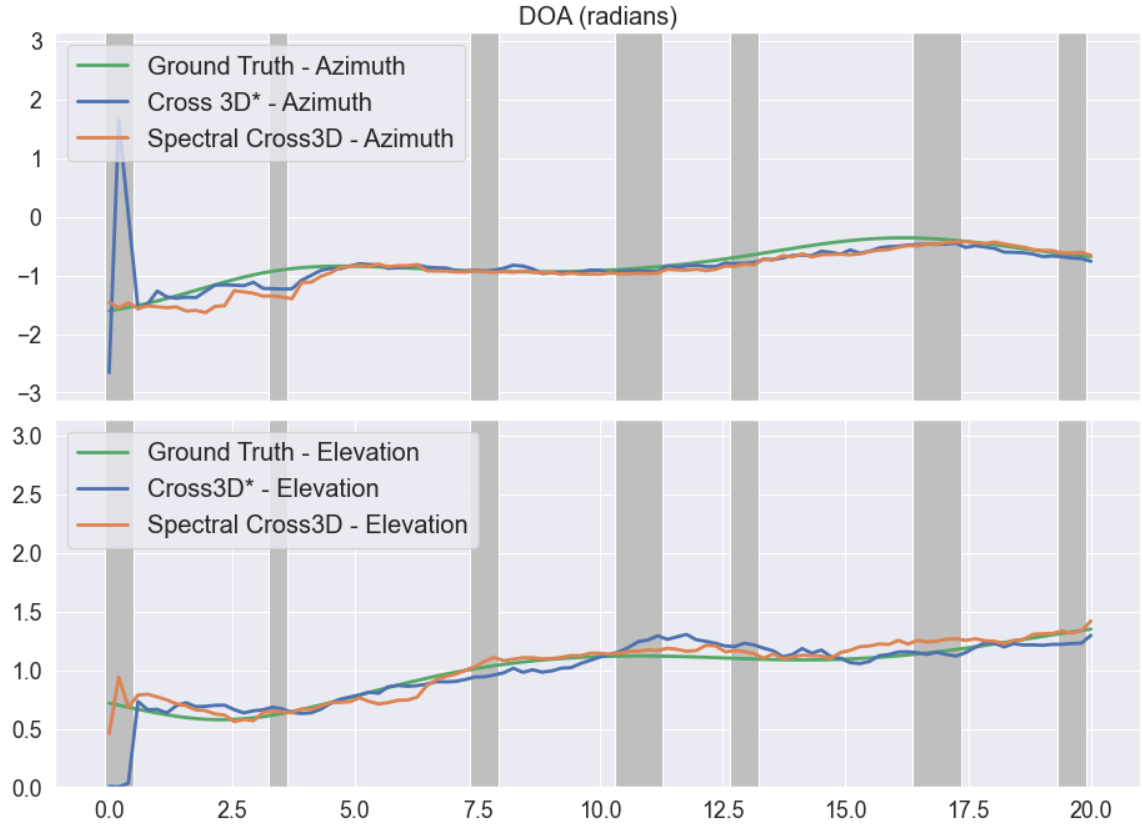


Figure 6.7: Example of C3D* (in blue) and SpC3D_1 (in orange) in an acoustic scene, where the ground truth is shown in green. The grey areas represent the silent frames identified by the VAD.

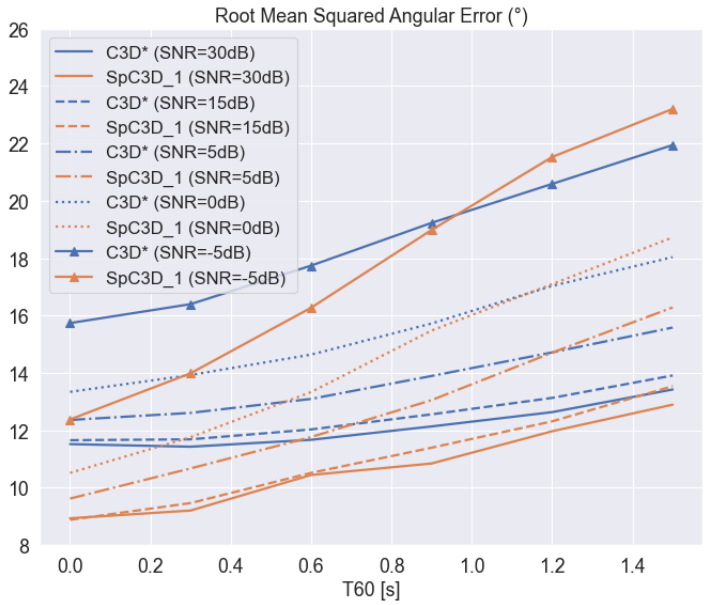


Figure 6.8: Comparison of the RMSAE attained with the retrained Cross3D (blue) and the Spectral Cross3D with the full spectrum (orange), for different values of Reverberation time (x-axis) and for different SNR levels (different traces).

moving speakers and moving microphone arrays.

The mean absolute azimuth error (MAAE), adopted as the comparison metric in the challenge, was calculated for SpC3D_1 and compared with the values corresponding to the LOCATA baseline (BL) in Table 6.3. As can be seen, the results show that SpC3D_1 yields a performance comparable to that of the BL. Even in Task 5, which proposes a scenario on which our system has not been trained, our results outperform the baseline.

Task	MAAE (SpC3D)	MAAE (BL)
Task 1	6.6°	4.2°
Task 3	4.5°	9.4°
Task 5	4.2°	5.4°

Table 6.3: Mean Absolute Azimuthal Error in the LOCATA Challenge dataset.

Chapter 7

Conclusions

In this work, we have introduced Spectral Cross3D, a new system for sound source tracking based on deep learning and the MUSIC DoA estimation method. It is inspired by a state-of-the-art architecture, called Cross3D [52], with major modifications in the pre-processing chain and some minor changes both in the neural architecture and in the training procedure. Initially, a comparison was drawn using our system with different preprocessing stages, and the use of the full spectrum has proven to be best solution amongst the tested approaches. With this setup, we have shown that our system was able to achieve results on average superior to those obtained with Cross3D (even the latter taking advantage of our remodeled training stage): 9.1% better in terms of the RMSAE in a test dataset. We also observed that a viable result was attained in the LOCATA Challenge dataset, in terms of the MAAE measure.

7.1 Future Works

Although many different criteria have been tried for the spectral selection mechanism, not only can others be proposed and evaluated, but also a more extensive parameter evaluation can be performed. Even though the present work suggests that the full spectrum solution is more successful in the tracking task, understanding how these choices affect the overall system performance is of extreme importance to moving forward.

Another aspect that must be addressed in the continuation of this work is the simulator software. Although gpuRIR provides the necessary speed for performing simulations simultaneously with the network training, it uses a method that does not model phenomena such as frequency-varying reflection coefficients. Training with a more realistic simulator could better characterize the performance of systems in a real usage scenario.

Finally, it is worth mentioning that, while this work was being concluded, other

solutions inspired by Cross3D were proposed [66, 67], pushing the state-of-the-art to even higher standards. Testing whether these systems would benefit from the preprocessing approach presented in this work should be another important research direction.

References

[1] BENESTY, J., CHEN, J., HUANG, Y. *Microphone Array Signal Processing*. Heidelberg, Germany, Springer, 2008. ISBN: 978-3-540-78611-5.

[2] DIBIASE, J. H., SILVERMAN, H. F., BRANDSTEIN, M. S. "Robust Localization in Reverberant Rooms". In: Brandstein, M., Ward, D. (Eds.), *Microphone Arrays: Signal Processing Techniques and Applications*, Springer, pp. 157–180, Berlin, Germany, 2001. ISBN: 978-3-662-04619-7. doi: 10.1007/978-3-662-04619-7_8.

[3] ZHANG, B., MASAHIDE, K., LIM, H. "Sound source localization and interaction based human searching robot under disaster environment". In: *2019 SICE International Symposium on Control Systems (SICE ISCS)*, pp. 16–20, Kumamoto, Japan, March 2019. doi: 10.23919/SICEISCS.2019.8758766.

[4] TAN, T.-H., LIN, Y.-T., CHANG, Y.-L., et al. "Sound source localization using a convolutional neural network and regression model", *Sensors*, v. 21, n. 23, December 2021. ISSN: 1424-8220. doi: 10.3390/s21238031.

[5] ABU-EL-QURAN, A., GOUBRAN, R., CHAN, A. "Security monitoring using microphone arrays and audio classification", *IEEE Transactions on Instrumentation and Measurement*, v. 55, n. 4, pp. 1025–1032, 2006. doi: 10.1109/TIM.2006.876394.

[6] WANG, W., LI, J., HE, Y., et al. "Symphony: Localizing Multiple Acoustic Sources with a Single Microphone Array". In: *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20, p. 82–94, New York, USA, 2020. Association for Computing Machinery. ISBN: 9781450375900.

[7] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning*. Cambridge, United States, MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] DMOCHOWSKI, J. P., BENESTY, J. "Steered Beamforming Approaches for Acoustic Source Localization". In: Cohen, I., Benesty, J., Gannot, S.

(Eds.), *Speech Processing in Modern Communication: Challenges and Perspectives*, Springer, pp. 307–337, Berlin, Germany, 2010. ISBN: 978-3-642-11130-3. doi: 10.1007/978-3-642-11130-3_12.

[9] KUTTRUFF, H. *Room Acoustics*, v. 1. 6 ed. Heidelberg, Germany, Springer, 2008. ISBN: 9780367870997.

[10] KAY, S. *Intuitive Probability and Random Processes using MATLAB®*. New York, United States, Springer, 2006.

[11] DINIZ, P. S. R., DA SILVA, E. A. B., NETTO, S. L. *Digital Signal Processing: Systems analysis and design*. 2 ed. New York, United States, Cambridge, 2010. ISBN: 978-0521887755.

[12] KNAPP, C., CARTER, G. "The generalized correlation method for estimation of time delay", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 24, n. 4, pp. 320–327, August 1976. doi: 10.1109/TASSP.1976.1162830.

[13] ROTH, P. R. "Effective measurements using digital signal analysis", *IEEE Spectrum*, v. 8, n. 4, pp. 62–70, April 1971. doi: 10.1109/MSPEC.1971.5218046.

[14] AL-HUSSAINI, E., KASSAM, S. "Robust Eckart filters for time delay estimation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 32, n. 5, pp. 1052–1063, October 1984. doi: 10.1109/TASSP.1984.1164428.

[15] CARTER, G., NUTTALL, A., CABLE, P. "The smoothed coherence transform", *Proceedings of the IEEE*, v. 61, n. 10, pp. 1497–1498, October 1973. doi: 10.1109/PROC.1973.9300.

[16] DONOHUE, K. D., HANNEMANN, J., DIETZ, H. G. "Performance of phase transform for detecting sound sources with microphone arrays in reverberant and noisy environments", *Signal Processing*, v. 87, n. 7, pp. 1677–1691, July 2007. ISSN: 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2007.01.013.

[17] HUANG, Y. A., BENESTY, J., CHEN, J. "Time Delay Estimation and Source Localization". In: Benesty, J., Sondhi, M. M., Huang, Y. A. (Eds.), *Springer Handbook of Speech Processing*, Springer, pp. 1043–1063, Berlin, Germany, 2008. ISBN: 978-3-540-49127-9. doi: 10.1007/978-3-540-49127-9_51.

[18] SCHAU, H., ROBINSON, A. "Passive source localization employing intersecting spherical surfaces from time-of-arrival differences", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 35, n. 8, pp. 1223–1225, August 1987. doi: 10.1109/TASSP.1987.1165266.

[19] OZEKI, K. *Theory of Affine Projection Algorithms for Adaptive Filtering*. Tokyo, Japan, Springer, 2015. ISBN: 9784431557371.

[20] CAPON, J. "High-resolution frequency-wavenumber spectrum analysis", *Proceedings of the IEEE*, v. 57, n. 8, pp. 1408–1418, August 1969. doi: 10.1109/PROC.1969.7278.

[21] GRIFFITHS, L., JIM, C. "An alternative approach to linearly constrained adaptive beamforming", *IEEE Transactions on Antennas and Propagation*, v. 30, n. 1, pp. 27–34, January 1982. doi: 10.1109/TAP.1982.1142739.

[22] BERGER, M., SILVERMAN, H. "Microphone array optimization by stochastic region contraction", *IEEE Transactions on Signal Processing*, v. 39, n. 11, pp. 2377–2386, November 1991. doi: 10.1109/78.97993.

[23] LIMA, M. V. S., MARTINS, W. A., NUNES, L. O., et al. "A volumetric SRP with refinement step for sound source Localization", *IEEE Signal Processing Letters*, v. 22, n. 8, pp. 1098–1102, December 2015. doi: 10.1109/LSP.2014.2385864.

[24] GAO, S., HUANG, Y., ZHANG, T., et al. "A Modified Frequency Weighted MUSIC Algorithm for Multiple Sound Sources Localization". In: *IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pp. 1–4, Shanghai, China, November 2018. doi: 10.1109/ICDSP.2018.8631636.

[25] YANG, J.-M., CHOI, M.-S., KANG, H.-G. "Two-channel DOA estimation usign frequency selective MUSIC algorithm with a phase compensation in reverberant room". In: *5th IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 365–368, Darmstadt, Germany, July 2008. doi: 10.1109/SAM.2008.4606891.

[26] KRIM, H., VIBERG, M. "Two decades of array signal processing research: the parametric approach", *IEEE Signal Processing Magazine*, v. 13, n. 4, pp. 67–94, July 1996. doi: 10.1109/79.526899.

[27] VIBERG, M., OTTERSTEN, B., KAILATH, T. "Detection and estimation in sensor arrays using weighted subspace fitting", *IEEE Transactions on Signal Processing*, v. 39, n. 11, pp. 2436–2449, November 1991. doi: 10.1109/78.97999.

[28] ŽMOLÍKOVÁ, K., DELCROIX, M., KINOSHITA, K., et al. "Learning speaker representation for neural network based multichannel speaker extraction". In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 8–15, Okinawa, Japan, December 2017. doi: 10.1109/ASRU.2017.8268910.

[29] NERCESSIAN, S., LUKIN, A. "Speech dereverberation using recurrent neural networks". In: *22nd International Conference on Digital Audio Effects (DAFx-19)*, pp. 152–156, Birmingham, United Kingdom, September 2019.

[30] ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T. *Learning From Data*. California, United States, AMLBook, 2012. ISBN: 1600490069.

[31] BISHOP, C. *Pattern Recognition and Machine Learning*. New York, United States, Springer, 2013. ISBN: 978-0-387-31073-2.

[32] MACK, W., BHARADWAJ, U., CHAKRABARTY, S., et al. "Signal-aware broadband DOA estimation using attention mechanisms". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4930–4934, Barcelona, Spain, May 2020. doi: 10.1109/ICASSP40776.2020.9053658.

[33] MANE, S. S., MALI, S. G., MAHAJAN, S. P. "Localization of steady sound source and direction detection of moving sound source using CNN". In: *10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, Kanpur, India, July 2019. doi: 10.1109/ICCCNT45670.2019.8944612.

[34] ADAVANNE, S., POLITIS, A., VIRTANEN, T. "Direction of arrival estimation for multiple sound sources using convolutional recurrent neural network". In: *26th European Signal Processing Conference (EUSIPCO)*, pp. 1462–1466, Rome, Italy, September 2018. doi: 10.23919/EUSIPCO.2018.8553182.

[35] DING, J., REN, B., ZHENG, N. "Microphone array acoustic source localization system based on deep learning". In: *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 409–413, Taipei, China, November 2018. doi: 10.1109/ISCSLP.2018.8706599.

[36] TANG, Z., KANU, J. D., HOGAN, K., et al. "Regression and Classification for Direction-of-Arrival Estimation with Convolutional Recurrent Neural Networks". In: *Proc. Interspeech 2019*, pp. 654–658, Graz, Austria, September 2019. doi: 10.21437/Interspeech.2019-1111.

[37] PEROTIN, L., DÉFOSSEZ, A., VINCENT, E., et al. "Regression Versus Classification for Neural Network Based Audio Source Localization". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 343–347, New Paltz, USA, October 2019. doi: 10.1109/WASPAA.2019.8937277.

[38] SALVATI, D., DRIOLI, C., FORESTI, G. L. "Exploiting CNNs for Improving Acoustic Source Localization in Noisy and Reverberant Conditions", *IEEE Transactions on Emerging Topics in Computational Intelligence*, v. 2, n. 2, pp. 103–116, March 2018. doi: 10.1109/TETCI.2017.2775237.

[39] WANG, Z.-Q., ZHANG, X., WANG, D. "Robust Speaker Localization Guided by Deep Learning-Based Time-Frequency Masking", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 27, n. 1, pp. 178–188, October 2019. doi: 10.1109/TASLP.2018.2876169.

[40] GRUMIAUX, P.-A., KITIĆ, S., GIRIN, L., et al. "A Survey of Sound Source Localization with Deep Learning Methods", *The Journal of the Acoustical Society of America*, v. 1, pp. 107–151, July 2021. doi: 10.1121/10.0011809.

[41] TAKEDA, R., KOMATANI, K. "Sound source localization based on deep neural networks with directional activate function exploiting phase information". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 405–409, Shanghai, China, March 2016. doi: 10.1109/ICASSP.2016.7471706.

[42] THRUN, S., BURGARD, W., FOX, D., et al. *Probabilistic Robotics*. Cambridge, United States, MIT Press, 2005. ISBN: 9780262201629.

[43] KOCH, W. *Tracking and Sensor Data Fusion*. Berlin, Germany, Springer, 2014.

[44] FÉ, J., CORREIA, S. D., TOMIC, S., et al. "Kalman filtering for tracking a moving acoustic source based on energy measurements". In: *International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pp. 1–6, October 2021. doi: 10.1109/ICECCME52200.2021.9590919.

[45] CHEN, C., WANG, H., ALI, A., et al. "Particle filtering approach to localization and tracking of a moving acoustic source in a reverberant room". In: *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, v. 4, Tolouse, France, May 2006. doi: 10.1109/ICASSP.2006.1661102.

[46] DO, H., SILVERMAN, H. F. "Stochastic particle filtering: A fast SRP-PHAT single source localization algorithm". In: *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 213–216, 2009. doi: 10.1109/ASPAA.2009.5346540.

[47] ELMAN, J. L. "Finding Structure in Time", *Cognitive Science*, v. 14, n. 2, pp. 179–211, March 1990. doi: https://doi.org/10.1207/s15516709cog1402_1.

[48] JORDAN, M. I. "Serial Order: A Parallel Distributed Processing Approach". In: Donahoe, J. W., Dorsel, V. P. (Eds.), *Neural-Network Models of Cognition*, v. 121, North-Holland, pp. 471–495, Amsterdam, Netherlands, 1997. doi: https://doi.org/10.1016/S0166-4115(97)80111-2.

[49] HOCHREITER, S., SCHMIDHUBER, J. "Long Short-term Memory", *Neural computation*, v. 9, pp. 1735–80, December 1997. doi: 10.1162/neco.1997.9.8.1735.

[50] CHO, K., VAN MERRIËNBOER, B., BAHDANAU, D., et al. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, out. 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012.

[51] LEA, C., FLYNN, M. D., VIDAL, R., et al. "Temporal Convolutional Networks for Action Segmentation and Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1003–1012, July 2017. doi: 10.1109/CVPR.2017.113.

[52] DIAZ-GUERRA, D., MIGUEL, A., BELTRAN, J. R. "Robust sound source tracking using SRP-PHAT and 3D convolutional neural networks", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 29, pp. 300–311, November 2021. doi: 10.1109/TASLP.2020.3040031.

[53] PEEBLES, JR., P. Z. *Probability, Random Variables and Random Signal Principles*. 4 ed. New York, United States, McGraw-Hill, 2001. ISBN: 978-0071181815.

[54] HAYES, M. *Statistical Digital Signal Processing and Modeling*. New York, United states, Wiley, 1996. ISBN: 9780471594314.

[55] LÖLLMANN, H., MOORE, A., NAYLOR, P., et al. "Microphone Array Signal Processing for Robot Audition". In: *IEEE Workshop on Hands-free Speech*

Communication and Microphone Arrays, pp. 51–55, San Francisco, United States, March 2017. IEEE Signal Processing Society, IEEE. doi: 10.1109/HSCMA.2017.7895560.

[56] PANAYOTOV, V., CHEN, G., POVEY, D., et al. "Librispeech: An ASR corpus based on public domain audio books". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, Brisbane, Australia, April 2015. doi: 10.1109/ICASSP.2015.7178964.

[57] DIAZ-GUERRA, D., MIGUEL, A., BELTRAN, J. R. "gpuRIR: A python library for room impulse response simulation with GPU acceleration", *Multimedia Tools and Applications*, v. 80, n. 4, pp. 5653–5671, February 2021. ISSN: 1573-7721. doi: 10.1007/s11042-020-09905-3.

[58] ALLEN, J. B., BERKLEY, D. A. "Image method for efficiently simulating small-room acoustics", *The Journal of the Acoustical Society of America*, v. 65, n. 4, pp. 943–950, 1979. doi: 10.1121/1.382599.

[59] LEHMANN, E. A., JOHANSSON, A. M. "Diffuse Reverberation Model for Efficient Image-Source Simulation of Room Impulse Responses", *IEEE Transactions on Audio, Speech, and Language Processing*, v. 18, n. 6, pp. 1429–1439, 2010. doi: 10.1109/TASL.2009.2035038.

[60] TORRES, J. C. "BRASS - Brazilian Room Acoustic Simulator". In: *XXVIII ENCONTRO DA SOBRAC*, Porto Alegre, Brazil, January 2018. doi: 10.17648/sobrac-87152. (in portuguese).

[61] RINDEL, J. H. "Computer Simulation Techniques for Acoustical Design of Rooms", *Acoustics Australia / Australian Acoustical Society*, v. 23, pp. 81–86, 01 1995.

[62] WISEMAN, J. "py-webrtcvad". `https://github.com/wiseman/py-webrtcvad`, 2022. [Online; accessed 17-May-2022].

[63] KELLS, L. M., KERN, W. F., BLAND, J. R. *Plane and Spherical Trigonometry*. New York, United States, McGraw-Hill, 1940. ISBN: 9781296235581.

[64] KINGMA, D. P., BA, J. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, May 2015.

[65] EVERS, C., LÖLLMANN, H., MELLMANN, H., et al. "The LOCATA Challenge: Acoustic Source Localization and Tracking", *IEEE/ACM Trans-*

actions on Audio, Speech, and Language Processing, April 2020. doi: 10.1109/TASLP.2020.2990485.

[66] ZHONG, T., VELÁZQUEZ, I. M., REN, Y., et al. "Spherical convolutional recurrent neural network for real-time sound source tracking". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5063–5067, Marina Bay Sands, Singapore, May 2022. doi: 10.1109/ICASSP43922.2022.9747845.

[67] DIAZ-GUERRA, D., MIGUEL, A., BELTRAN, J. R. "Direction of Arrival Estimation of Sound Sources Using Icosahedral CNNs". Available at `https://arxiv.org/abs/2203.16940`, 2022.